Summer 8-31-1998

Requirement elicitation and knowledge management utilizing relational database, client server and internet technologies

Umang J. Dave
New Jersey Institute of Technology

ABSTRACT

The first step in any project is a statement of requirements. Requirements specification is an inevitable part of any successful project. However, the "rush" to convert concepts to products often proves as a big hindrance in the development process of any requirements set. This, in turn, hampers the ability to produce the concept and manage knowledge as originally envisioned.

The goal of the thesis is to foster understanding among the different communities affected by the development of the given system. The thesis is based on the principle that the requirements elicitation process for complex system is fundamentally a conversation among the stakeholders that is designers, customers and implementers. The web based tool developed for requirements elicitation allows the stakeholders to pool their respective expertise and viewpoints to resolve requirement issues. This leads to consensus building among the stakeholders and also provides with well-defined, clear, and concise requirements set. The requirements gathered by the tool can help determine the source, applicability, depth, and other factors needed for assessing and implementing integrated, and coherent, requirements set. It also helps the design team identify whether a specific requirement establishes a quantifiable threshold. Moreover, the tool uses the latest technologies of client server architecture, relational database and Internet. This make the tool efficient, portable, easy to debug. The three tier architecture of the tool also facilitates the ease in enhancement.

The thesis also emphasizes that research efforts should be directed towards methods and tools needed to improve requirements analysis process and in particular to those providing more support to the elicitation of requirements. A simple lesson that can be learnt is that no one person knows everything about what a system should do. There are always many participants in a successful requirement elicitation effort.

REQUIREMENT ELICITATION AND KNOWLEDGE MANAGEMENT UTILIZING RELATIONAL DATABASE, CLIENT SERVER AND INTERNET TECHNOLOGIES

by
Umang J. Dave

A Thesis
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Computer Science

Department of Computer and Information Science

August 1998

APPROVAL PAGE

REQUIREMENT ELICITATION AND KNOWLEDGE MANAGEMENT UTILIZING RELATIONAL DATABASE, CLIENT SERVER AND INTERNET TECHNOLOGIES

Umang J. Dave

8/24/98

Dr. Murat Tanik, Thesis Advisor — Date
Department of Computer and Information Science
New Jersey Institute of Technology, Newark, NJ.

8/24/98

Dr. Ali Dogru, Committee Member — Date
Department of Computer and Information Science
New Jersey Institute of Technology, Newark, NJ.

8-24-98

Dr. Franz Kurfess, Committee Member — Date
Department of Computer and Information Science
New Jersey Institute of Technology, Newark, NJ.

8/24/98

Leon Jololian, Committee Member — Date
Department of Computer and Information Science
New Jersey Institute of Technology, Newark, NJ.

BIOGRAPHICAL SKETCH

Author: Umang J. Dave

Degree: Master of Science in Computer Science

Date: August 1998

Undergraduate and Graduate Education:

- Master of Science in Computer Science,
  New Jersey Institute of Technology, NJ, 1998

- Bachelor of Science in Computer Engineering,
  Gujarat University, Ahmedabad, India, 1994.

Major: Computer Science

Dedicated to my mother, father and my beautiful wife – Jigna.

ACKNOWLEDGEMENT

I would like to express my greatest appreciation to Prof. Murat Tanik without whom the Requirements Elicitation project would have been only another good idea. I also thank him for providing me with an excellent opportunity to work with him and for his valuable guidance. I want to express my gratitude and thank the committee members of my thesis – Prof. Ali Dogru, Prof. Franz Kurfess and Prof. Leon Jololian for their continuous cooperation and guidance. I also, thank Prof. Marcus Healey for his inspiration.

I would like to thank the Graduate Studies department at New Jersey Institute of Technology and Mrs. Annette Damiano for helping me with my thesis documentation.

I take this opportunity to thank Merrill Lynch for funding the project. I thank the computer science department and New Jersey Institute of Technology for providing me with resources and support.

Very special thanks are due to my friends Raj, Mariam and rest of my friends who have helped me at various stages of my thesis.

Finally, I am grateful to my mother, father and my wife for their faith, prayers and understanding during the many hours that this thesis took my time from the, This is especially true of my wife, Jigna, whose patience and understanding fulfilled my need for sustenance and inspiration.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

LIST OF FIGURES

LIST OF TABLES

Table **Page**

CHAPTER 1

INTRODUCTION

Requirements Engineering is more a social activity than a precise technical activity. Nevertheless, it is an important and necessary aspect of software engineering, and one that helps distinguish software engineering from computer science.

1.1 Requirements Engineering

Software requirements negotiation is the process where the customers' needs in a software project are identified. This process is regarded as one of the most important parts of building a software system because during this stage it is decided precisely what will be built. Requirements negotiation is an iterative process where, through reflection and experience, users become familiar with the technology and developers become familiar with the user needs [HERLEA 97].

To produce quality products, understanding of requirements for a software system is a major concern. Requirement engineering is a common terminology used to specify various requirements related activities. Requirement engineering comprises of four specific processes [RAGHAVAN 94] :

1. Requirement Elicitation - The process, through which customers, buyers, or users of a software system discover, reveal, articulate, and understand their requirements. Experience over the last 30 years that incorrect, incomplete, or misunderstood requirements are the most common causes of poor quality, cost overruns, and late delivery of software systems. The ability to employ a systematic process for

requirements elicitation is therefore one of the fundamental skills of a good software engineer [RAGHAVAN 94].

2. Requirements analysis - The process of reasoning about the requirements that have been elicited; it involves activities such as examining requirements for conflicts or inconsistencies, combining related requirements, and identifying missing requirements [RAGHAVAN 94].
3. Requirements specification - The process of recording the requirements in one or more forms, including natural language and formal, symbolic, or graphical representations; also, the product that is the document produced by that process [RAGHAVAN 94].
4. Requirements validation - The process of confirming with the customer or user of the software that the specific requirements are valid, correct, and complete [RAGHAVAN 94].

The software requirement elicitation process is an essential activity to the development of quality software products. A framework has been developed by Micheal Christel and Kyo Kang, members of the Software Engineering Institute (SEI), located at Carnegie Mellon University in Pittsburgh, to address the requirement elicitation process [MILLER 93]. The requirement elicitation process model consists of:

1. Fact-Finding Phase: This process allows an examination of the organization into which the target system will be placed, develops high level statements of the system's missions and roles, determines constraints on the architecture of the system's mission and roles, determines constraints on the architecture, and identifies the existence of the similar systems [CHRISTEL 92].

2. Requirement Gathering and Classification: This phase allows the gathering and organizing of information with the help of multiple views which express the information that is to be built [CHRISTEL 92].
3. Evaluation and Rationalization: Rationalization and evaluation phase is responsible for exposing inconsistencies in the gathered requirements and determining why the information has been expressed as a requirement [CHRISTEL 92].
4. Prioritization: This phase determines the relative importance of each requirement and the relative order the requirements should be addresses in [CHRISTEL 92].
5. Integration and Validation: This phase combines all the information acquired in the proceeding phases and creates a set of requirements. Validation is performed to determine that the requirements meet the goals and objectives outlined during the fact-finding stage [CHRISTEL 92].

1.2 Knowledge Management

Knowledge management entails formally managing knowledge resources in order to facilitate access and reuse of knowledge, typically by using advanced information technology. A wide range of technologies are being used to implement knowledge management systems : e-mail, databases and data warehouses, group support systems, browsers and search engines, internets and intranets, expert and knowledge-based systems, and intelligent agents [DANIEL 98].

Since earlier days knowledge has been stored on papers and in people's minds. Such a storage leads to inefficient management of knowledge as it is extremely difficult to maintain and update. Storing data in knowledge warehouses and knowledge bases

greatly improves the management, maintenance and updating of data. Moreover, data stored in such a form has a wide accessibility.

It is also absolutely necessary for the user as well as the developers to understand the environment in which the system is supposed to function. The requirements must be specified such that the system can easily be integrated into the other existing systems. The lack of domain knowledge accessible by the user or developer can lead to improper knowledge management.

Gregory Piatetsky-Shapiro and William Frawley define knowledge discovery as "non-trivial extraction of implicit, previously unknown, and potentially useful information from data. Because knowledge discovery approaches can be designed to exploit characteristics and structures of the underlying application domain, knowledge discovery has found use in a wide range of applications.

Knowledge bases can become quite large and have great amount of information. Searching them efficiently becomes an extremely critical function. The most dominant search techniques include search engines, intelligent agents, and visualization models [DANIEL 98].

1.3 Environmental Life Cycle

A primary thrust of industrial ecology is that manufacturers practice product stewardship - designing, building, maintaining, and recycling products in such a way that they pose minimal impact to the wider world. Product stewardship should be broadly interpreted to include services, which should also be performed so as to have minimal impact. The way in which these tasks are addressed in formal manner is by the process of Life-Cycle

Assessment (LCA), a family of methods for looking at materials, services, products, processes, and technologies over their entire life [GRAEDEL 95].

The essence of life-cycle assessment is the evaluation of the relevant environmental, economic, and technological implications of a material, process, or product across its life span from creation to waste or, preferably, to re-creation in the same or another useful form [GRAEDEL 95].

LCA is made up of three stages: inventory analysis, impact analysis, and improvement analysis, as pictured in Fig. 1. First, the scope of the LCA is defined. An inventory analysis and an impact analysis are then performed, the result being an environmentally responsible product rating (R_{ERP}). This rating guides an analysis of potential improvements (which may feed back to influence the inventory analysis). Finally, the improved product is released for manufacture [GRAEDEL 95].

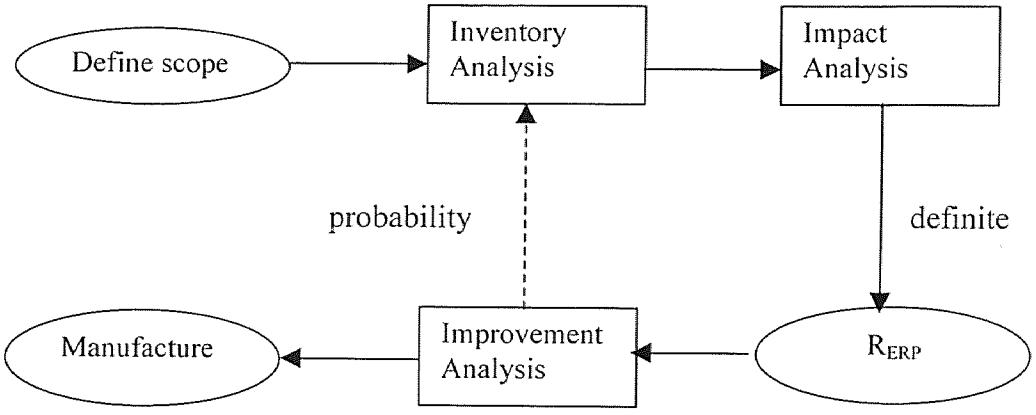

Figure 1 Steps in the Life-Cycle Assessment of a Product. R_{ERP} is the environmentally responsible product rating. [based on GRAEDEL 95.]

An effective LCA technology must be able to quickly and easily identify, then differentiate between, critical environmental impacts. This will allow designers to concentrate on the most important problems, reserving for later those that produce lesser impacts [GRAEDEL 95].

According to Robert Ayres of the European Institute of Business Administration the work done on materials at the expense of energy represents society's battle against thermodynamics, and the energy invested per unit of material decreases as one approaches the top of the materials flow chain.

The generality of the options set should also be explicitly defined. In many cases, it is possible to generate numerous potential options, not all of which can be feasibly reviewed. The challenge is to choose representatives options that provide valuable guidance for real-world decisions, but are limited and general enough to make an analysis realistic [GRAEDEL 95.].

The conclusion then is "If it mandates that something must be accomplished, transformed, produced, or provided, it is a requirement – period". The qualifiers follows – in the form of characteristics and relationships. This allow a requirement to be the focus, i.e. the key player or authority, for all related information in a requirements management database environment [HARWELL 93]

CHAPTER 2

STANDARD TECHNOLOGIES AND METHODOLOGIES

Studying user needs is a first step to any solution, along with gaining an understanding of available technologies and existing tools. These two tasks interact. Without an understanding of technologies one may aim for the impossible, and without an understanding of needs, one may solve the wrong problem [BERLIN 89].

By far, the most common kind of requirements elicitation effort is one that gets information directly from the people who will use the system. In such cases, the elicitation procedure can be described in very general terms as five steps [RAGHAVAN 94]:

1. Identify relevant sources of requirements (the users).
2. Ask them appropriate questions to gain an understanding of their needs.
3. Analyze the gathered information, looking for implications, inconsistencies, or unresolved issues.
4. Confirm your understanding of the requirements with the users.
5. Synthesize appropriate statements of the requirements.

2.1 Knowledge Management

Cooperative work systems such as world wide web and Lotus Notes are beginning to tackle the aspect of knowledge management. Database systems can contribute much of the required functionality. Hence it is required to integrate functionality and ideas from these sources. With the web and the use of search engines, many people are already experiencing significant changes in how they use and manage knowledge [SKUCE 97].

2.1.1 Knowledge Representation

Knowledge Management systems represent knowledge in both human and machine readable forms. Human-readable knowledge is typically accessed using browsers or intelligent search agents. But some knowledge is accessible for machine-readable purposes, designed as an expert system's knowledge base to support decision making [DANIEL 98].

1. Human Readable Knowledge: Case-specific information appears to provide the appropriate level of representation required for users to make best use of the knowledge. Where the information is largely declarative knowledge text or rules might be used to represent the information and knowledge. If on the other hand, information is highly filtered, then it is likely to be represented as a set of declarative statements [DANIEL 98].
2. Logic: The most popular way of representing knowledge is by formal logic. Formal logic provides a very powerful tool for software development and can be used in all stages of the development process [PEYMAN 97].
3. Machine Readable Knowledge: Expert systems use their knowledge bases and user responses to guide the user to recommended solutions. Expert systems can be an integral part of knowledge management system. Although some knowledge management systems contain such artificial intelligence-based systems, most knowledge management systems use artificial intelligence primarily in the form of intelligent agents to search human-readable knowledge [DANIEL 98].
4. Frames: Frames are used for declarative knowledge. Frames were developed because there is evidence that people subjected to new situations do not analyze the situation

from scratch, but analyze situations using previous experience from objects, other people, locations, and so on. Frames contain information about many aspects of an object or an action. A frame might for example contain slots with necessary characteristics of an object, possible characteristics and typical instances of that object. Frames can be used in situations where there is a large amount of context dependent knowledge [PEYMAN 97].

5. Scripts: Scripts are clusters of facts about typical sequence of events in a given context. They can have a set of entry conditions, i.e. a set of conditions that must be met before the script can be executed. Scripts can be very useful in expressing events with very casual relationships. Scripts can also provide a way of dividing events into sub-events. Scripts can provide powerful tools in software development. Scripts can provide some support for reuse in design and the means for discovering and compensating the absence of a certain type of information in the requirements. They can be used to record facts about an existing software system, by keeping track of and organizing information about interaction between the systems components. They can also be used to describe the object interaction in a software model by describing normal flow of interaction between objects [PEYMAN 97].
6. Semantic Nets: Semantic nets are networks used to express semantic structures. The structure is very simple. They are built of nodes connected by arcs. There are objects and there are relationships between these objects [PEYMAN 97].

2.1.2 Knowledge Filtering

Systems often resort to knowledge filtering to ensure complete and correct knowledge. Not all filtering is done by humans. Perhaps the most visible and frequent use of computer-based filtering is the message filtering that categorizes and prioritizes e-mail messages. A number of products also help monitor qualitative databases [DANIEL 98].

2.1.3 Knowledge Search

1 Search Engines : A wide range of well-known internet search engines - like AltaVista, Excite, Infoseek, Lycos, WebCrawler, and Yahoo - have been used to guide users to information on the Internet. These and other search engines can be adapted to intranet environment for knowledge management. Alternative approaches to conventional search engines are also developed [DANIEL 98].

2 Intelligent agents : Intelligent agents can be used to connect people to knowledge available on the Internet or intranets. Heuristics can be used to gather additional insights into user's interests. Based on message syntax, attempts can be made to determine significant phrases that provide insight into user goals [DANIEL 98].

3 Visualization models : Two emerging tools - Perspecta and InXight - represent different ways of visualizing knowledge space. Perspecta creates what it calls SmartContent using meta information derived from source documents - be it structured information in databases and tagged documents such as news feeds, or unstructured information in office documents and Web pages. For unstructured documents , it has a Document Analysis Engine that performs linguistic analysis and automatically tags documents. InXight software, a spin-off from Xerox PARC,

recently released its VizControl information visualization software for visualizing large hierarchies. VizControl technology offers several novel visualization formats, each of which exploit "focus + context" techniques that foreground objects of interest while preserving the overall structure of even very large data sets [DANIEL 98].

2.2 Software Engineering Institute's Requirement Engineering Framework

According to the Draft Pamphlet (Draft PAM, April 23, 1991) by Requirements Definition Implementation Team, of Software Test and Evaluation Panel, requirements engineering is "the disciplined application of scientific principles and techniques for developing, communicating and managing requirements". Loucopoulos and Champion define requirements engineering as "the systematic process of developing requirements through an iterative process of analyzing a problem, documenting the resulting observations and checking the accuracy of the understanding gain."

Requirements elicitation normally involves several developers (the requirements analysts and software engineers) and several customers (the buyers or users of the software). Each of these persons brings different knowledge and skills to the effort [RAGHAVAN 94].

There are three learning cycles, as shown in Figure 2. The developers are assisted by the users in gaining new viewpoints about their requirements, and through reformulating the requirements, the user learns more about them. The system receives pressure for evolution as the users learn more about how it can be used, and the system induces that learning on the users. The system evolves by actions of the developers, who

in turn gain enhanced understanding of the system through that evolution [RAGHAVAN 94].

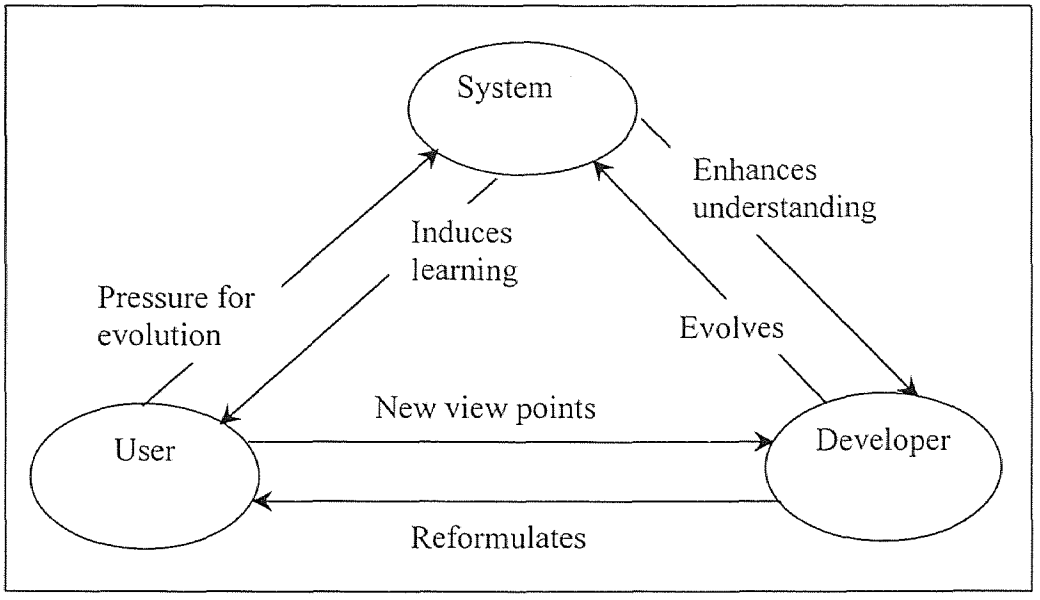

Figure 2 Adaptive loops learning cycles [based on RAGHAVAN 94].

The requirements elicitation process using the adaptive loops framework focuses on addressing, supporting, and facilitating these learning cycles. It is especially useful when there are requirements articulation problems, and it is helpful in overcoming some of the technical issues of requirements elicitation for the evolution of complex systems [RAGHAVAN 94].

2.2.1 Requirements Elicitation Framework

Rzepka decomposes the requirements engineering process into three activities [RZEPKA 89]:

- elicit requirements from various individual sources;

- insure that the needs of all users are consistent and feasible; and
- validate that the requirements so derived are an accurate reflection of user needs.

Elicitation will likely iterate with these activities during requirements development.

Prior to SEI's proposed requirements framework, the requirements elicitation process consisted of scattered processes, methodologies and techniques : each having their own benefits, strengths, and weaknesses. By themselves, none of these strategies were able to address all the attributes necessary to produce a quality set of requirements that met the needs of the stakeholder community. Existing models, methodologies, and techniques failed to address the problems inherent in the requirements elicitation activity namely : system scope, understanding among participants of the process, understanding among personnel effected by the process, and a recognition of the volatility of requirements [MILLER 93].

In an effort to address these issues, the framework was established. It is composed of a process model, methodology, and a group of supporting techniques. The process model guides the methodology and techniques imposed by the framework. A methodology is a fine grained activity that supports a process. Methodologies are prescriptive in nature and often recommend methods, techniques, and tools to enact them [MILLER 93].

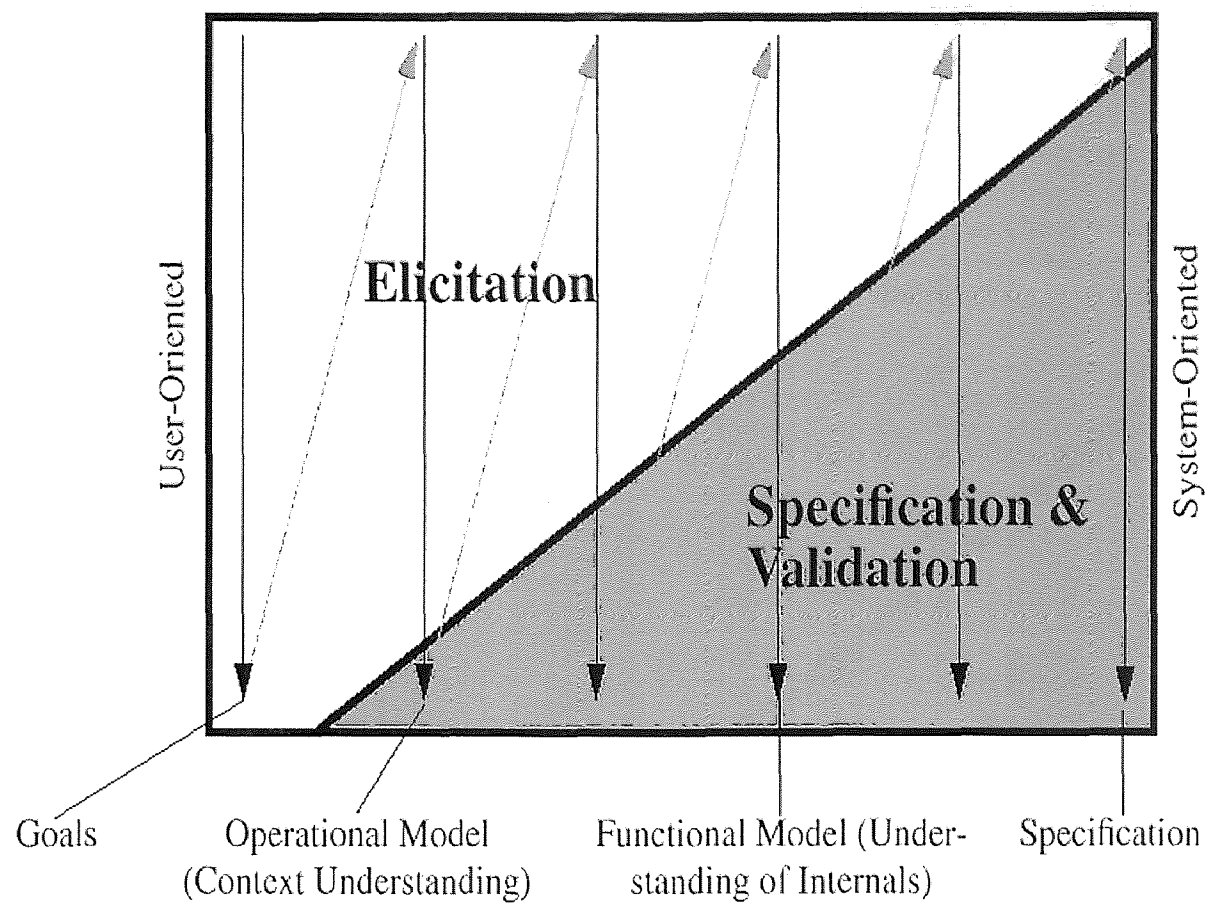

Figure 3 Requirements Engineering is an Iterative Process [CHRISTEL 92].

One of the integral part of the requirement elicitation framework is formal specification language. A formal specification language offers [PLAYLE 96]:

1. Clarity : Formal Specification languages can remove some elements of ambiguity from the process. They offer explicit syntax and semantics that define the language and a set of relations that precisely defines object interaction .
2. Consistency : Because the language is relatively fixed, a formal specification language reduces the chance for misinterpretation when it passes through various development groups and lifecycle phases .

3. Completeness : Formal specification languages are incomplete, and some languages are purposely incomplete to allow designers some freedom. Developers and designers must guard against incompleteness to ensure all the requirements are treated consistently .
4. Prototyping : Prototyping, as a means of requirements elicitation, can be highly effective . Tamai and Itou also found that during the prototyping phase users were not intimately familiar with the system, so developers provided a simplified interface. As the users became more familiar with the system, they demanded more functionality from the software, resulting in a more complicated interface. When this product was delivered, the users required both the simplified version and the complicated version. The conclusion was that "features" and "interim requirements" may become part of the users' expectations and eventually become system requirements .

2.2.2 Requirement Elicitation Process Model

The communication between the user-oriented and developer-oriented activities is cyclical, and enhanced via modeling. The communication enhancement is desirable. The representation of the requirements should promote understanding while allowing for inevitable change, and hence this representation should be introduced as early into the requirements engineering process as possible while still maintaining the desirable characteristics of modifiability (extensibility and evolvability), readability, and analyzability [CHRISTEL 92].

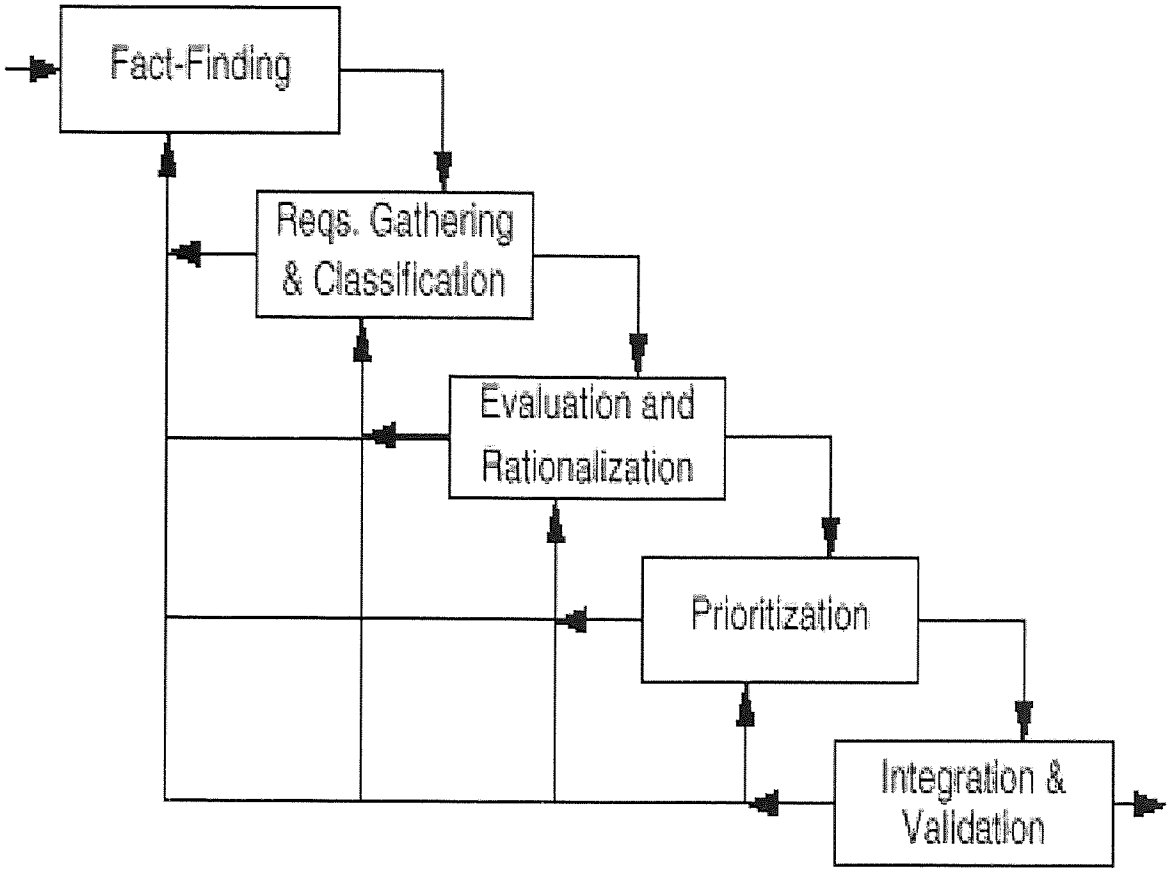

Figure 4 Requirement Elicitation Process Model [CHRISTEL 92].

This elicitation process model is first executed during the concept exploration phase of system development, which is initiated after the creation of a mission needs statement. Following this phase, the first level of detail in the requirements specification is achieved. During the subsequent demonstration and validation phase, these specifications are validated, the unclear requirements clarified, the unknown requirements identified, and the existing ones refined as necessary. Based on communication mechanisms (such as prototyping) employed during this phase, these elicitation steps are then cycled through again beginning with "requirements gathering"

to detail and improve the requirements document. The demonstration and validation phase is entered with incomplete requirements, and therefore that these elicitation process steps are returned to after the first pass through the concept exploration phase [CHRISTEL 92].

With regard to the user community, fact-finding begins with identifying the relevant parties at multiple levels within the community, e.g., from a high-level commander for a strategic long term perspective to an end user for the immediate perspective. The operational context and problem context are defined, perhaps through goal trees and mission statements, which help with the later filtering of the requirements. This includes an objectives analysis, which studies the user organization's objectives, constraints against full achievement of the objectives, and their influences and interactions. Context analysis and the determination of operational modes and mission scenarios completes the user-oriented task fact-finding activities. The developer oriented fact-finding tasks are performed in parallel [CHRISTEL 92].

2.2.3 Requirement Engineering Techniques

The requirements definition process comprises these steps : [BRACKETT 90]

- Requirements Identification,
- Identification of software development constraints,
- Requirements analysis,
- Requirements representation,
- Requirements communication and
- Preparation for validation of software requirements

Techniques for requirements elicitation generally provide operational-level tactics and guidelines. They usually focus narrowly on specific aspects of the elicitation process. Such techniques are [RAGHAVAN 94] :

1. Brainstorming : Brainstorming is a simple group technique for generating ideas. It allows people to suggest and explore ideas in an atmosphere free of criticism or judgment. The session consists of two phases. In the generation phase, participants are encouraged to offer as many ideas as possible, without discussion of the merits of the ideas. In the consolidation phase, the ideas are discussed, revised, and organized .

   For purposes of software requirements elicitation, brainstorming can be helpful in generating a wide variety of views of the problem and in formulating the problem in different ways. It is especially useful very early in the elicitation process. When used correctly, it can help overcome some of the underlying difficulties of requirements elicitation [RAGHAVAN 94]:

   - It stimulates imaginative thinking to help users become aware of their needs,
   - It helps build a more complete picture of the users' needs,
   - It can avoid the tendency to focus too narrowly too soon and
   - For some personality types, it provides a more comfortable social setting than some of the more structured group techniques.

   Good brainstorming sessions are very helpful in overcoming some of the cognitive limitations of participants by allowing (or forcing) them to expand their thinking. The lack of criticism and judgment during the generation phase also helps overcome some of the communication barriers of requirements elicitation [RAGHAVAN 94].

2. Interviewing : Interviewing is an important technique for eliciting detailed information from an individual. It is commonly used in requirements elicitation for large systems as part of some of the high level elicitation techniques. It can also be used for small projects as the only requirements elicitation technique [RAGHAVAN 94].

   Interviewing is not simply a matter of asking question. It is a more structured technique that can be learned, and software engineers can gain proficiency with training and practice. It requires the development of some general social skills, the ability to listen, and knowledge of a variety of interviewing tactics [RAGHAVAN 94].

   A skilled interviewer can help the user to understand and explore software requirements, thus overcoming many of the articulation problems and communications barriers [RAGHAVAN 94].

2.2.4 Requirement Elicitation Methodology and its Methods

Most software engineering methods presume that requirements are explicitly and completely stated; however, experience shows that requirements are rarely complete and usually contain implicit requirements [PLAYLE 96].

The elicitation methodology should be prescriptive in nature in order to provide the guidance as to how the specifications should be elicited from the user. Guidelines for tailoring the methodology to specific problems will most likely be developed, validated, and refined iteratively. As the methodology matures and more problem areas are addressed, the framework will grow as well [CHRISTEL 92].

2.2.4.1 Fact Finding: The very first step in requirements elicitation involves determining what is the problem to be addressed, and who needs to be involved in this decision-making as well as who will be affected by the problem's formulation and eventual solution. The output from this activity includes [CHRISTEL 92]:

- a statement of the problem context,
- the overall objectives of the target system and
- boundaries and interfaces for the target system

Activities performed in this phase are the creation of operational, problem, and organizational contexts, identification and documentation of similar systems, and the assessment of cost and implementation constraints imposed by the customer [CHRISTEL 92].

If the understanding and the representation of the problem domain is mature, the objectives of this phase may be easily identified, understood, and completed. However, if this is not the case, cross functional teams should be engaged to perform this task. The objective of cross functional teams is to involve experts to aid in the definition and development of the outputs for this stage, and to involve the relevant parties in order to create a sense of commitment and shared ownership. The accuracy and completeness of this phase is critical to the success of the entire process. Multiple passes through this phase should be considered to promote completeness [MILLER 93].

An effective approach to achieving this cross-disciplinary communication for fact-finding is the use of a group process technique, such as Joint Application Design (JAD). All the affected parties should be represented in the group which will perform these early fact-finding tasks. This promotes shared ownership, rapid early problem

formulation, and an aligned perspective and understanding between the elicitation communities of the problem to be solved and the scope of the subsequent requirements [CHRISTEL 92].

2.2.4.2 Gathering and Classification: It is important to gather as much information as possible from users, developers, and customers. Some of this information may come from the group development techniques employed during fact-finding, such as JAD. More information can be gathered through the use of interviews directly with end users and other affected parties. Questionnaires, observations, and simulation environments are other techniques that can be utilized to get information from different individuals and groups. The output from this activity includes [CHRISTEL 92]:

- customer and user oriented objectives and
- customer and user oriented needs and requirements

A tailored JAD session serves to provide the framework for this step in the requirements elicitation process. Structured interviews and questionnaires are used to capture and document information and underlying rationale. The JAD process starts with a problem-research phase. Analysts perform structured interviews to identify the relevant parties and to acquire the information and rationale necessary to meet the goals and objectives of this phase. The structure and the format of the interviews is dictated by the techniques used to model and represent the information required [MILLER 93].

The views are better understood if they can be structured into manageable pieces. This is especially true given that the elicitation process will be incremental, to deal with inevitable changes in requirements. If we return to monolithic views of the complete

system, it will be very difficult to both comprehend such a large view and also to find portions of that view which may be affected by an incremental change to the requirements. Thus, there must also be a decompositional process associated with requirements gathering, where the views can be broken down into meaningful components [CHRISTEL 92].

2.2.4.3 Evaluation and Rationalization: The goal of this phase is to fully develop and evaluate the underlying rationale behind the requirements gathered to this point. A risk assessment should be performed to address technical, cost, and schedule concerns. In addition, the rationale behind the information gathered in previous stages should be examined to determine whether the true requirements are hidden in this rationale instead of being explicitly expressed. This rationale and risk assessment are the two main outputs from this activity [CHRISTEL 92].

The objectives, goals, and constraints developed in the fact-finding phase are compared to the requirements detailed in this documents and models developed in requirements gathering and classification phase. The evaluations are performed by the requirements analyst with the aid of the stakeholders through a series of structured interviews [MILLER 93].

The evaluation is performed by comparing the requirements representations against the organizational, goal, and constraint models. The comparison determines if the requirements address the right issues [MILLER 93].

The rationale process is performed in parallel with the prior evaluation. This consists of a series of interviews in which the analyst and the stakeholders evaluate the

requirements model against the rationale stored. The objective is to determine why each requirement is present [MILLER 93].

Once the rationale has been collected and examined, inconsistencies can ideally be found and better choices on decision points or issues made to both resolve these inconsistencies and address the needs reflected in the rationale. In addition, this rationale is extremely useful in later passes through the elicitation methodology as documentation on why particular choices were made. If incremental changes to the requirements are to be made, these changes can be checked to see if they are in agreement with the rationale underlying the rest of the existing requirements [CHRISTEL 92].

2.2.4.4 Prioritization and Planning: The goal of the prioritization phase is to arrange the requirements in order of relative importance from the view of the client and the view of the developer [CHRISTEL 92].

Incremental development, of both the system and the requirements, is stressed in the process model. If requirements are prioritized, then high priority needs can be addressed first, and the subsequent requirements changes defined and reexamined, before low priority needs (which could change as well) are implemented. This can result in cost and time savings when processing the inevitable requirements changes during system development. The requirements must be prioritized based on cost, dependency, and user needs [CHRISTEL 92].

2.2.4.5 Integration and Validation: The goal of the integration and validation is to reduce the conflicts found in the requirements, to address completeness, and to validate the requirements [CHRISTEL 92].

Outputs of this activity are a set of complete requirements or a set of incomplete but validated requirements ready for the specification and formal validation processes. Output may also be in the form of requirements which are lacking in some quality. The incomplete requirements are iterated through the requirements elicitation model to resolve the open issues [MILLER 93].

Integration of multiple views should occur as much as possible through the involvement of all the affected communities, so that this shared ownership is not lost. Validation of the requirements by all affected parties ensures that their concerns are met. Subsequent passes through the elicitation methodology outlined here address the requirements deficiencies, inconsistencies, and other problems found during the demonstration and validation steps [CHRISTEL 92].

The DoD software technology plan states that "early defect fixes are typically to orders of magnitude cheaper than late defect fixes, and the early requirements and design defects typically leave more serious operational consequences." One way to reduce requirements error is by improving requirements elicitation [CHRISTEL 92].

CHAPTER 3

DOMAIN AND PROBLEMS

Requirements problems have been identified as major contributors to program cost overruns and schedule slips. This "problem with requirements" can be attributed to two major factors [IVY 90] :

1. The initial requirements were incomplete, inaccurate, and / or misunderstood by the designers / developers.
2. Circumstances changed which resulted in changes to the requirements.

3.1 Problems in Requirements Engineering

There are many problems associated with requirements engineering, including problems in defining the system scope, ensuring understanding between developer and user, and problems in dealing with the volatile nature of requirements. These problems may lead to poor requirements and the cancellation of system development, or else the development of a system that is later judged unsatisfactory or unacceptable, has high maintenance costs, or undergoes frequent changes. Issues involved in this problem area include [CHRISTEL 92].:

- Achieving requirement completeness without unnecessarily constraining system design,
- Analysis and validation difficulty and
- Changing requirements over time

Software requirements characteristically suffer from inconsistency, incompleteness, nonspecific language (ambiguity), duplication, and inconstancy. On a complex system, the documents that make up the requirements are voluminous [PLAYLE 96].

Cooperation between participants in the process is quite difficult, even if they meet in a physical meeting room. The process involves a social network of people with different professional backgrounds and different views over the system that must be built. If the participants in the process are in different organizations or different cities, meetings can be costly, inconvenient and infrequent. This leads to problems of communication, which can greatly impact the quality of the elicited requirements [HERLEA 97].

Problems of requirements elicitation can be grouped into three categories: [CHRISTEL 92].

1 problems of scope, in which the requirements may address too little or too much information;
   - the boundary of the system is ill-defined and
   - unnecessary design information may be given

2 problems of understanding, within groups as well as between groups such as users and developers;
   - users have incomplete understanding of their needs,
   - users have poor understanding of computer capabilities and limitations,
   - analysts have poor knowledge of problem domain,
   - user and analyst speak different languages,
   - ease of omitting "obvious" information,

- conflicting views of different users and
- requirements are often vague and untestable, e.g., "user friendly" and "robust"

3 problems of volatility, i.e., the changing nature of requirements.

- requirements evolve over time

3.1.1 Problems of Scope

Elicitation techniques need to be broad enough to establish boundary conditions for the target system, yet still should focus on the creation of requirements [CHRISTEL 92].

Requirements elicitation must begin with an organizational and context analysis to determine the boundary of the target system as well as the objectives of the system. Less ambitious elicitation techniques not addressing this concern run the risk of producing requirements which are incomplete and potentially unusable, because they do not adhere to the user's or organization's true goals for the system. Performing an organizational and context analysis allows these goals to be captured, and then used to verify that the requirements are indeed usable and correct [CHRISTEL 92].

The problem is manifested by the fact that the requirements focus is often in what is to be built without consideration for how it will be operation. Operations costs are directly driven by program definition and the requirements imposed by all elements of the program. Frequently, the requirements do not consider the operations concepts, constraints, or plan. When operations requirements are finally considered, addition requirements will be needed to overcome this initial oversight [IVY 90].

Elicitation techniques can be overly ambitious as well. Elicitation must focus on the creation of requirements, not design activities, in order to adequately address users' concerns and not just developers' needs [CHRISTEL 92].

Environmental factors have a strong influence on requirements elicitation. Environmental factors include: [CHRISTEL 92].

- hardware and software constraints imposed on a target system (the target system will typically be a component of some larger system with an existing or required architecture already in place),
- the maturity of the target system's domain,
- the certainty of the target system's interfaces to the larger system and
- the target system's role within a larger system

3.1.2 Problems of Understanding

Problems of understanding can be separated into three issues: [CHRISTEL 92]

- The communities involved in elicitation possess a variety of backgrounds and experience levels, so that which is common knowledge to one group may be completely foreign to another. This makes it difficult for a requirements analyst to interpret and integrate information gathered from these diverse communities.
- The language used to express the requirements back to these stakeholder communities may be too formal or too informal to meet the needs of each of the groups, again because of the diversity of the communities.

- The large amount of information gathered during elicitation necessitates that it be structured in some way. The understanding of this structure is dependent on the characteristics of the stakeholder communities.

3.1.3 Problems of Volatility

Requirements change. During the time it takes to develop a system the users' needs may mature because of increased knowledge brought on by the development activities, or they may shift to a new set of needs because of unforeseen organizational or environmental pressures. If such changes are not accommodated, he original requirements set will become incomplete, inconsistent with the new situation, and potentially unusable because they capture information that has since become obsolete [CHRISTEL 92].

One primary cause of requirements volatility is that "user needs evolve over time". The requirements engineering process of elicit, specify, and validate should not be executed only once during system development, but rather should be returned to so that the requirements can reflect the new knowledge gained during specification, validation, and subsequent activities. A requirements engineering methodology should be iterative in nature, "so that solutions can be reworked in the light of increased knowledge" [CHRISTEL 92].

3.2 Problems in Knowledge Management

Knowledge is a critical resource but we still do not have many new ideas on how to manage it. Most knowledge is currently kept in conventional documents that are hard to structure, classify, browse, search, and even find. Most of the knowledge is still recorded

as unstructured natural language, with all its shortcomings of precision and conciseness. Documents in particular, small parts of them such as sentences, are often hard or impossible to find, hard to update cooperatively, and hard to keep coordinated merging of information from various sources is difficult. Most systems today offer nothing in the way of inference, semantic checking, or natural language processing. Finding and organizing documents is very difficult [SKUCE 97].

For frames to be used in a formal manner, there needs to be some standard ways of representing them. There needs to be a specific number of slots in each frame, or type of frame. [PEYMAN 97]. Even though semantic nets were developed a long time ago, except for very basic relationships there are no standard methods of how the arcs should be labeled. It should be mentioned that even the most basic facts have problems associated with them [OBJA 95].

3.3 Problems in Environmental Life Cycle

In college and university computer science programs, instructors usually create the requirements specification; students are rarely involved in the process. It is even more rare for students to be taught the specific techniques that software engineers use for requirements elicitation. This can probably be attributed to the absence of these techniques from most computer science textbooks and the lack of familiarity with these techniques on the part of instructors [RAGHAVAN 94].

The problems in requirement engineering, knowledge management and environmental life cycle cite that the software tools build for them are not used at all, or are used by a small minority of people. The most basic reason of this is that the end users

were not involved while developing such tools. It is very difficult to build a successful system without end user involvement.

CHAPTER 4

RELATIONAL DATABASE AND INTERNETTECHNOLOGIES FOR REQUIREMENT ELICITATION

The hardest single part of building a software system is deciding precisely what to build. No other part of the conceptual work is as difficult as establishing the detailed technical requirements, including all the interface to people, to machines, and to other software systems. No part of the work so cripples the resulting system if done wrong. No other part is more difficult to rectify later [BROOKS 87].

4.1 Relational Database Management Systems

A database-management system (DBMS) consists of a collection of interrelated data and a set of programs to access those data. Database is referred to as collection of information related to a particular enterprise. The most important use of a database is to provide an environment that is both convenient and efficient to use in retrieving and storing database information.

4.1.1 Requirement of Database Systems

Today's environment contains humungous amount of information. In order to manage all this large information, database systems were originated. This management of data involves both the definition of structures for the storage of information and provision of mechanisms for the manipulation of information [KORTH 97]. Along with this, database systems provide proper security measures to restrict unauthorized access of information.

Advantages of Database Systems are:

1. Eliminates data redundancy and inconsistency.
2. Facilitates easy access of data.
3. Provides data isolation.
4. Provides data integrity.
5. Facilitates concurrent-access of the database information.

4.1.2 Data Models

Data model is the underlying structure of any database system. It is used to describe the data, the data relationships, data semantics, and consistency constraints. The various data models are as described below:

1) Object-Based Logical Models

This model is used to describe data at the logical and view levels. They are characterized by the fact that they provide fairly flexible and structuring capabilities and allow data constraints to be specified explicitly [KORTH 97]. The various object-based logical models are:

2) The Entity-Relationship Model

The entity-relationship (E-R) data model is made up of collection of basic objects called *entities* and *relationships* among these objects. An entity is a real world "object" or "things" which has an independent existence. Each entity has its own specifications, which are indicated as *attributes* in the E-R schema. A relationship is an association between various entities of the E-R model.

The overall logical structure of a database can be expressed graphically by an E-R diagram. The components of the E-R schema are:

a) Rectangles: This represents the entity sets.

b) Ellipses: Represents attributes of the entities.

c) Diamonds: Represents relationship among entity sets.

d) Lines: Links attributes to entity sets and entity sets to relationships.

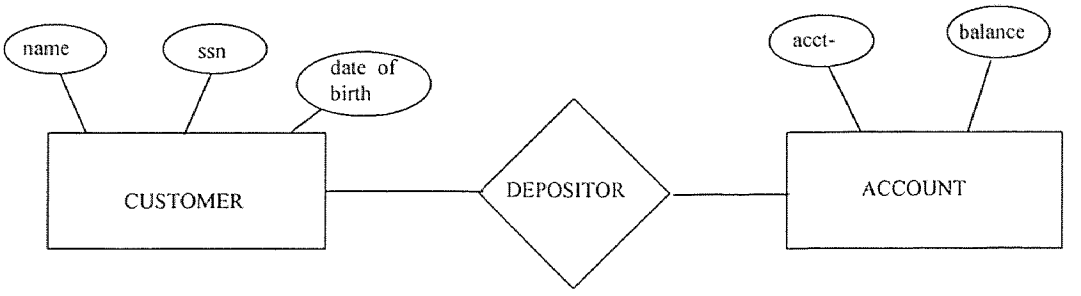

Figure 5 Sample ER Diagram

3) The Object-Oriented Model

This model is based on the collection of objects. An object contains values stored in instance variables within the object and the code, which defines the operation of that object. This code which defines the working of the object is called *methods*.

Objects that contain the same type of values and the same methods are grouped together into *classes* [KORTH 97]. Communication between objects is carried out with the help of *messages*. Object-oriented model isolates the variables and the methods defined in each object from another object, thus providing data encapsulation. The other features of object-oriented model are inheritance and polymorphism.

4) Relational Model

This model uses a collection of tables to represent both the data and the relationships among the data represented in the tables. Each tables is made up of multiples columns and rows. Each row of the table is called as the *tuple*.

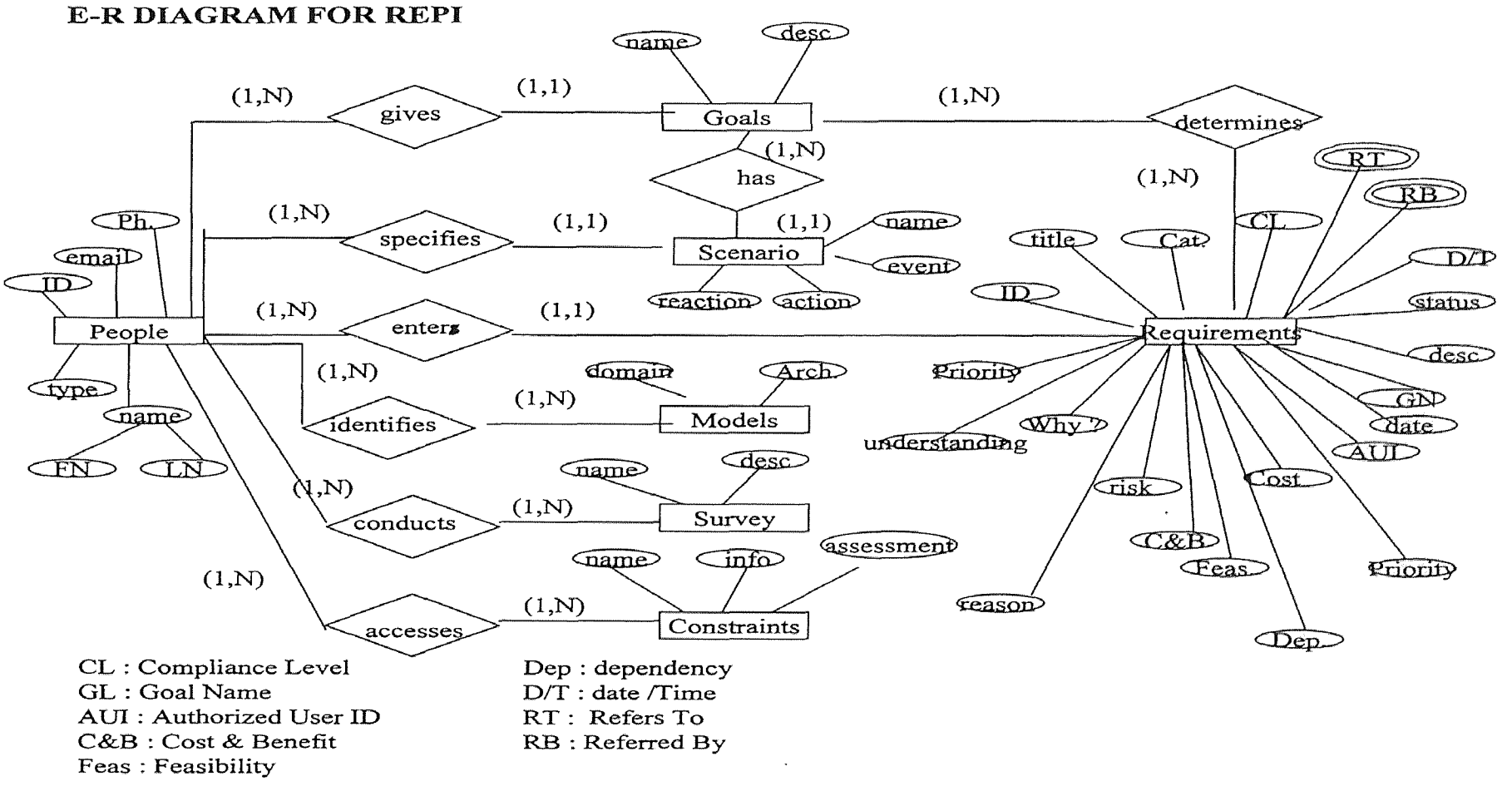

Figure 6 Entity Relationship Diagram

4.1.3 Database Languages

Database system provides two types of languages:

1. Specify the database schema - Data-Definition Language.

A database schema is specified by a set of definitions expressed by a special language called a data-definition language (DDL). The result of compilation of DDL statements is a set of tables that is stored in a special file called data dictionary or data directory [KORTH 97].

2. Express database queries and updates - Data-Manipulation Language.

This language allows:

a) Retrieval of information stored in the database.

b) Insertion of new information into the database.

c) Deletion of information from the database.

d) Modification of information stored in the database.

There are two types of data-manipulation languages:

1. Procedural DMLs requires a user to specify what data is required and how to retrieve that data.
2. Nonprocedural DMLs requires a user to specify what data is needed without specifying how to get that data.

4.1.4 System Structure

The relational database system structure is as shown in Figure 7.

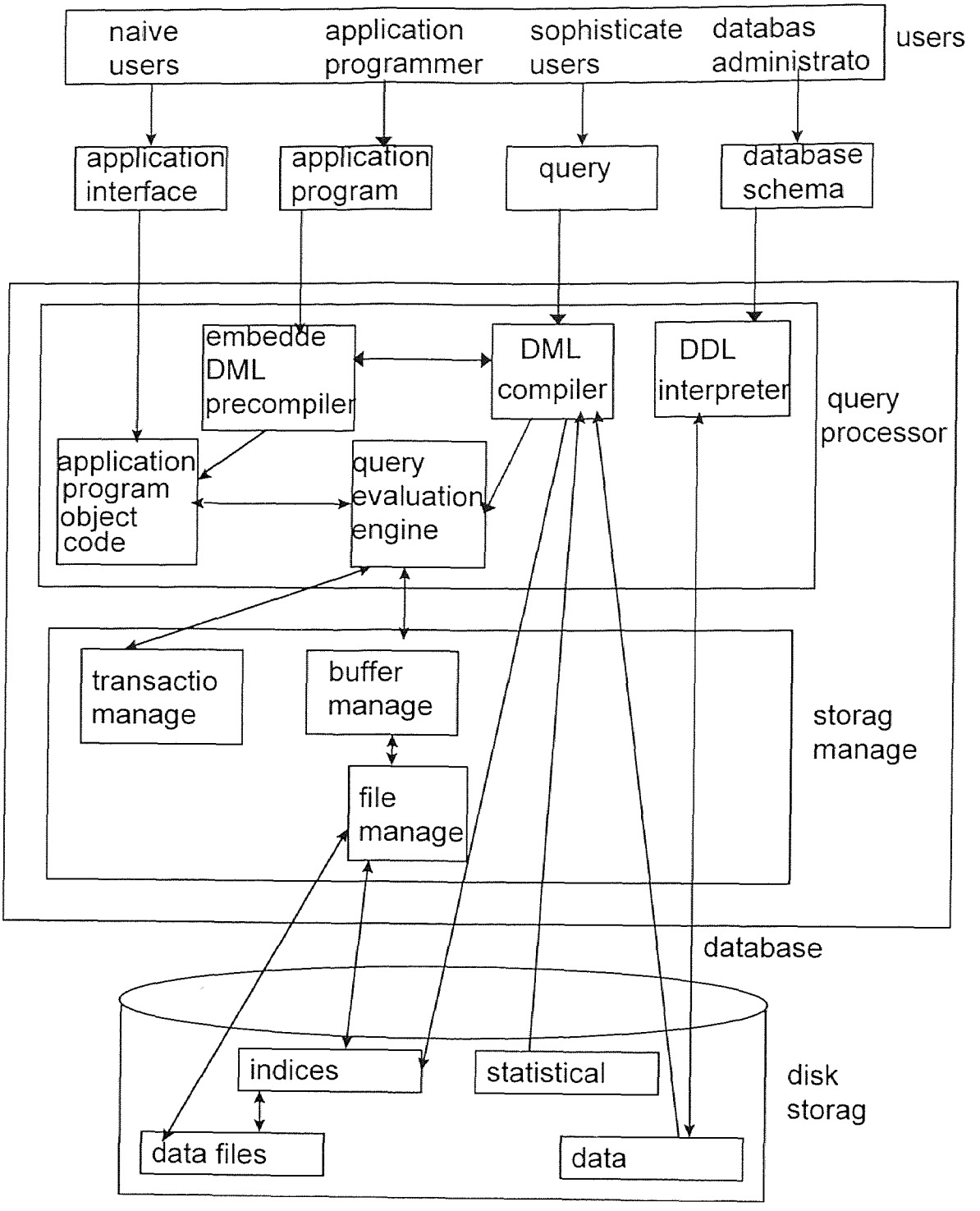

Figure 7 DBMS System Structure

4.1.5 Database Design for Requirement Elicitation

The database is designed by normalizing the tables that are used to store data. The design of the Requirement elicitation database and the normalized tables are shown in Table 1.

Table 1 Normalized table for Requirement Elicitation Database

Sr. No.	Table Name	Attribute Name	Type & Length	Key	Null/Not Null	Description
1	People_m	people_id	Char(11)	PK	NN	User/Developer Identification Number
		people_fname	Char(15)		NN	First name
		people_lname	Char(15)			Last name
		people_addr1	Char(20)			Address Line1
		people_addr2	Char(20)			Address Line 2
		people_email	Char(27)			Email
		people_phone	Char(12)			Phone Number
		people_type	Char(2)		NN	User or Developer
		people_desc	Char(200)			Description
		people_category	Char(2)			Category of developer or user
		people_filename	Char(20)			File name for information
2	Goal_m	goal_id	Char(11)	PK	NN	Goal Identification Number
		goal_pid	Char(11)	FK	NN	User/Developer Identification Number
		goal_desc	Char(200)			Goal Description
		goal_name	Char(40)		NN	Goal name
		goal_filename	Char(20)			File name for goal description
3	Req_m	req_id	Char(11)	PK	NN	Requirement Identification Number
		req_gid	Char(11)	FK	NN	Goal Identification #

Table 1 (continued) Normalized table for Requirement Elicitation Database

Sr. No.	Table Name	Attribute Name	Type & Length	Key	Null/Not Null	Description
		req_pid	Char(11)	FK	NN	User/Developer Identification Number
		req_title	Char(50)			Requirement Title
		req_desc	Char(200)			Requirement description
		req_category	Char(25)			Category
		req_valid	Char(1)			Validity of the requirement
		req_aui	Char(11)			Authorization Identification Number
		req_importance	Char(3)			Importance of requirement
		req_understanding	Char(3)			Understanding of requirement
		req_costlevel	Char(3)			Costlevel of requirement
		req_deplevel	Char(3)			Dependence level of requirement
		req_upriority	Char(3)			User priority of requirement
		req_dpriority	Char(3)			Developer priority of requirement
		req_complevel	Char(25)			Compliance level of requirement
		req_status	Char(25)			Status of requirement
		req_type	Char(2)			Requirement type
		req_filename	Char(20)			Filename to store requirement information
		req_why	Char(200)			Why the requirement is required
		req_reason	Char(200)			Reason of requirement
		req_risk	Char(200)			Risk involved in the requirement
		req_feasible	Char(200)			Feasibility of the requirement

Table 1 (continued) Normalized table for Requirement Elicitation Database

Sr. No.	Table Name	Attribute Name	Type & Length	Key	Null/Not Null	Description
		req_costben	Char(200)			Cost & benefit of the requirement
		req_dummy	Char(3)			Dummy variable required
4	Survey_m	survey_id	Char(11)	PK	NN	Survey Identification Number
		survey_pid	Char(11)	FK	NN	User/Developer Identification Number
		survey_name	Char(40)			Survey Name
		survey_desc	Char(200)			Survey description
		survey_filename	Char(20)			Filename to store survey information
5	Const_m	const_pid	Char(11)	FK	NN	User/Developer Identification Number
		const_name	Char(40)		NN	Constraint name
		const_info	Char(200)			Constraint information
		const_asses	Char(200)			Accessibility of the constraint
		const_filename	Char(20)			Filename to store constraint information
6	Model_m	model_pid	Char(11)	FK	NN	User/Developer Identification Number
		model_domain	Char(200)			Domain of the model
		model_arch	Char(200)			Architecture of the model
7	Scenario_m	scen_id	Char(11)	PK	NN	Scenario Identification Num.

Table 1 (continued) Normalized table for Requirement Elicitation Database

Sr. No.	Table Name	Attribute Name	Type & Length	Key	Null/Not Null	Description
		scen_gid	Char(11)	FK	NN	Goal Identification Number
		scen_pid	Char(11)	FK	NN	User/Developer Identification Number
		scen_name	Char(20)		NN	Scenario Name
		scen_desc	Char(200)			Scenario description
8	Scenario_det	scen_did	Char(11)	FK	NN	Scenario Identification Number
		scen_event	Char(25)			Scenario Event
		scen_action	Char(25)			Scenario Action
		scen_reaction	Char(25)			Scenario Reaction

4.2 Implementation

The web based tool for Requirement Elicitation consists of three major components.

- Client PC running web browser.
- Web server running html front pages, Java applets and application server for communicating with the database.
- Database server which stores all the information entered by various users and developers.

1. Client PC running web browser: The user as well as the developers interact with the system through the client PCs. The PC require a web browser. The web browser should be Netscape communicator 4.0 or higher. When the PC connects to the web

server using the web browser, the html pages gets downloaded on the PC and the user or developer can browse through the system using the browser.

2. Web server running Java applets and application server: Currently, the web server at New Jersey Institute of Technology is the server cache. It is running the apache web server at port 6001. The web server contains the html front pages and the Java applets. The Java applets are coded using Java script. The front pages as well as the applets gets downloaded on the client PC and run in the browser environment. There is a Java applet running behind each front page. The word documents that are stored on the server get downloaded on the PC, allowing the user to update the local copies. The web server also houses an application server, developed in Java. The application server is a generic server or a dispatcher that accepts the requests for database queries and then communicates with the database server. The applets initialize and opens a socket with application server, and all the communication between the applet and the application server is carried out through this socket. The application server uses JDBC to communicate with the database server. It then gets the reply from the database server, and passes it back to the applet, which eventually passes it to the front page. User then gets to see the results of his queries. The three generic operations supported by the application server are insert, update and retrieval from the database.
3. Database server: The database server that is used is the server logic at New Jersey Institute of technology. First of all an entity relationship diagram was developed for the system. Then the database tables were identified with their foreign keys and primary keys. The tables were then normalized giving the complete design of the

database. The relational database management system used for this project is Oracle 7. The tables were created in the appropriate format. The function of the database server is to serve to the requests submitted by the application server. The requests could be either an insert, update or a select request. The data stored in the database server is in a specific format. This data can then be downloaded to a plain text file with some identifiers. The data can also be uploaded directly onto other database and can be very easily used to generate report or presentations from it. The data if uploaded onto another database can even be used to generate more web pages from it and can be used as a benchmark with some other systems.

Currently, the tool is being used to gather information of various hazardous materials that are used in the manufacturing of weapons. The DoD is planning to use it to collect data related to these waste materials from various facilities. This data can then be available to the actual designers of the weapons and can be used in future design process or can be used for statistics for a forth coming project.

The procedure to be used for requirement elicitation using the tool is very straightforward. The application guides the user from one phase to another. The user starts from the fact-finding phase, gathering and classification phase, evaluation and rationalization phase, prioritization and planning phase and finally the integration and validation phase. He iterates thorough these phases.

In the fact-finding phase it is very important to identify the person entering the requirement, before one starts to enter requirements. The identification consists of entering personal information in the system. This is a mandatory step. The user then enters the problem specification. Entering the goal name and goal specification is also a

mandatory step, as the user might need this to associate the goal with the requirements. The user can also enter various mission scenario specifications as additional information. The developer must also identify domain experts. As additional information the developer can enter information related to technical surveys and assess constraints.

In the gathering and classification phase the user as well as the developer can enter or modify the information for a particular requirement. This is a mandatory step. The user as well as the developer can view the information entered for a particular requirement or can list all the requirements entered by the entire group. The developer also has the facility to classify requirements according to their category.

The evaluation and rationalization phase is used to abstract rationale from information entered. This provides additional information and can help in designing and planning. The user specifies why, and reason behind the requirement and the developer evaluates the risk, feasibility and cost / benefits of a particular requirement.

The user as well as the developer then prioritizes the requirements depending on the level of understanding, level of dependency, cost, importance and priority. This helps the developer to plan the incremental stages and design the system so that it addresses the most important issues first.

The integration and validation stage is used to ensure that the requirement is specified towards a goal and to obtain authorization. This phase ensure the completeness and consistency of the requirements.

The user as well as the developer iterate through these phases while specifying requirements and so can very speedily converge to a common platform. At this point the requirements are clear to both the parties and the system is well understood. The

developer can then proceed on to the design phase with a very clear picture of the requirements and its priorities.

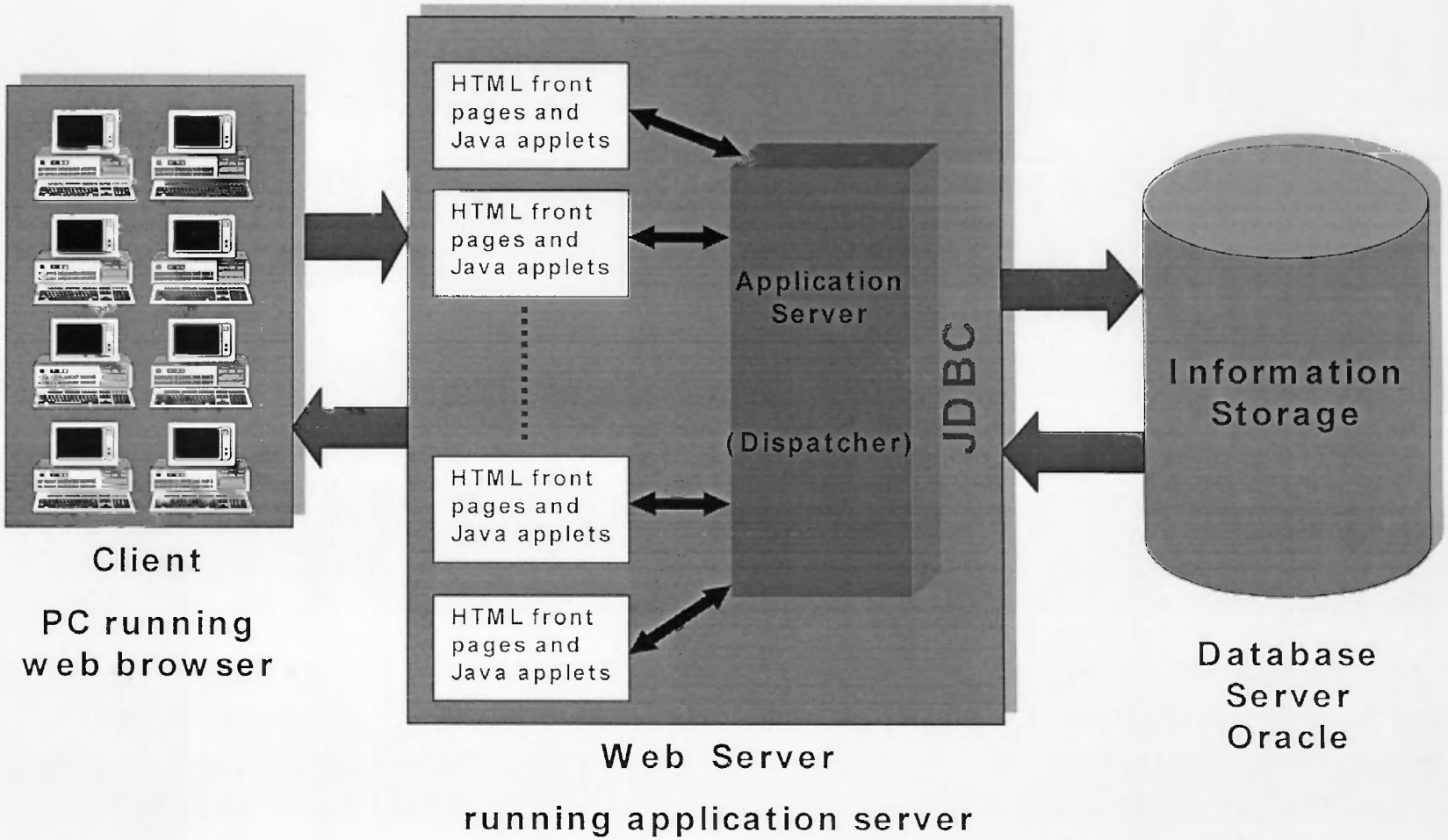

Figure 8 Requirement Elicitation System Design

Sample Source Code (Application Server (Dispatcher) - ApplicationServer.java) :

Generic Server for Database connectivity.

```
/**************************************************************************
<!-- REPI Web Site
<!-- Demo Version - August 14, 1998
<!-- Copyright (c) 1998
<!--
<!-- Author: Umang  Dave
<!--        Mariam  Burmawalla
<!-- Email: tanik@homer.njit.edu
**************************************************************************/
import java.awt.*;
import java.sql.*;
import java.lang.*;
import java.util.*;
import java.net.*;
import java.io.*;
class DBAdmin {
   private static final boolean DEBUG = true;
        private Connection con;
        private DatabaseMetaData dbmetad;
        private String stCatalog = " ";
        private Statement SQLstatement;
   public int nRowNum, nColNum;
        DBAdmin(String stUrl, String stName, String stPwd) {
      String stStatus;
                try {
        Class.forName("oracle.jdbc.OracleDriver");
        con = DriverManager.getConnection("jdbc:oracle:mariam/m15502@logic");
        dbmetad = con.getMetaData();
        stCatalog = con.getCatalog();
        SQLstatement = con.createStatement();
        stStatus = "Establishing a connection with a DataBase";
                }
                catch( Exception e ) {
                    stStatus = "Connection Failed" + e.getMessage();
           }
           if(DEBUG) System.out.println(stStatus);
        }

        public boolean fnExecSQLUpdate(String stQuery) {
                try {
        int nTmpVar = SQLstatement.executeUpdate(stQuery);
        return true;
```

```
    }
            catch(SQLException se) {
              if(DEBUG) System.out.println(se.getMessage());
            }
            return false;
        }


  private ResultSet fnExecSQLSelect(String stFieldName, String stTableName, String
stCondition){
            String stQuery = " ";
            ResultSet tmpres = null;
            try {
                    stQuery = fnstQueryBuilder("SELECT", stFieldName.trim(),
stTableName.trim(), stCondition.trim());
System.out.println("Say bye");
                    tmpres = SQLstatement.executeQuery(stQuery.trim());
System.out.println("Say bye1");
            }
            catch(SQLException se) {
                    if(DEBUG) System.out.println(se.getMessage());
                    if(DEBUG) System.out.println(se.getErrorCode());
            }
            return tmpres;
        }

  public String[][] fngetFieldsDB(String stFieldName, String stTableName, String
stCondition) {
            String stTmp = "";
            String stFieldVal[][];
            ResultSet tmpres;
            ResultSetMetaData tmpresmetdat;
            int nColumns = 0, nRows = 0;
            int i =0;
                    System.out.println("Bye1");
            tmpres = fnExecSQLSelect(stFieldName, stTableName, stCondition);
            if(tmpres == null){
            }
            else{
                    try{
                            tmpresmetdat = tmpres.getMetaData();
                            nColumns = tmpresmetdat.getColumnCount();
                    System.out.println("ByeBye");
                            if(nColumns >= 1){
                                    String stColumnName[] = new String[nColumns];
                                    String stColumnType[] = new String[nColumns];
```

```
                                    for(i=0;i<nColumns;i++){
                                        stColumnName[i] =
tmpresmetdat.getColumnName(i+1);

                                        stColumnType[i] =
tmpresmetdat.getColumnTypeName(i+1);

        System.out.println("Column type is "+stColumnType[i]);
                                    }
                                        while(tmpres.next()){
                                          nRows = nRows + 1;
                                            for(i=0;i<nColumns;i++){
                                    System.out.println("Column type again is
"+stColumnName[i]+stColumnType[i]);

        if(stColumnType[i].equals("VARCHAR2")){
System.out.println("Say Hi");
                                                    stTmp = stTmp +
tmpres.getString(stColumnName[i]) + "^";
                                                    //Each field value is separated
by a space
                                                }

        if(stColumnType[i].equals("LONG")){
                                                    stTmp = stTmp +
Long.toString(tmpres.getLong(stColumnName[i])) + "^";
                                                    //Each field value is separated
by a space
                                                }

        if(stColumnType[i].equals("BYTE")){
                                                    stTmp = stTmp +
Byte.toString(tmpres.getByte(stColumnName[i])) + "^";
                                                    //Each field value is separated
by a space
                                                }

        if(stColumnType[i].equals("SHORT")){
                                                    stTmp = stTmp +
Short.toString(tmpres.getShort(stColumnName[i])) + "^";
                                                    //Each field value is separated
by a space
                                                }

        if(stColumnType[i].equals("SINGLE")){
```

```
                                                    stTmp = stTmp +
Integer.toString(tmpres.getInt(stColumnName[i])) + "^";
                                                    //Each field value is separated
by a space
                                                }

        if(stColumnType[i].equals("DOUBLE")){
                                                    stTmp = stTmp +
Double.toString(tmpres.getDouble(stColumnName[i])) + "^";
                                                    //Each field value is separated
by a space
                                                }
                                            }
                                            if(stTmp.endsWith("^")){
                                stTmp = stTmp.substring(0,stTmp.length()-1);
                        }
                                            stTmp = stTmp.trim() + "%";
                                            //Each Tuple is separated by an underscore
                                    }
                                }
                                else{
                                    String stColumnType =
tmpresmetdat.getColumnTypeName(1);
                                    String stColumnName =
tmpresmetdat.getColumnName(1);
                                    while(tmpres.next()) {
                                      nRows = nRows + 1;
                                        if(stColumnType.equals("TEXT")) {
                                            stTmp =stTmp +
tmpres.getString(stColumnName) + "%";
                                            //Each field value is separated by a
space
                                        }
                                        if(stColumnType.equals("LONG")) {
                                            stTmp = stTmp +
Long.toString(tmpres.getLong(stColumnName)) + "%";
                                            //Each field value is separated by a
space
                                        }
                                        if(stColumnType.equals("BYTE")){
                                            stTmp = stTmp +
Byte.toString(tmpres.getByte(stColumnName)) + "%";
                                            //Each field value is separated by a
space
                                        }
                                        if(stColumnType.equals("SHORT")){
```

```
                                                stTmp = stTmp +
Short.toString(tmpres.getShort(stColumnName)) + "%";
                                                //Each field value is separated by a
space
                                        }
                                        if(stColumnType.equals("SINGLE")){
                                                stTmp = stTmp +
Integer.toString(tmpres.getInt(stColumnName)) + "%";
                                                //Each field value is separated by a
space
                                        }
                                        if(stColumnType.equals("DOUBLE")){
                                                stTmp = stTmp +
Double.toString(tmpres.getDouble(stColumnName)) + "%";
                                                //Each field value is separated by a
space
                                        }
                                }
                        }
                }
                catch(SQLException se){}
        }
        try {
                tmpres.close();
        }
        catch(SQLException se){}

        stTmp = stTmp.trim();
        if(stTmp.endsWith("%")){
                stTmp = stTmp.substring(0,stTmp.length()-1);
        }
    stFieldVal = new String[nRows][nColumns];
    stFieldVal = fnFieldParser(stTmp, nRows, nColumns);
        return stFieldVal;
    }
    private String fnstQueryBuilder(String stQueryType, String stFieldName, String
stTableName, String stCondition){
        String stQuery = " ";
        if(stQueryType == "SELECT"){
                if(stCondition == " "){
                   stQuery = "SELECT " + stFieldName + " FROM " +
stTableName;// + ";";
                }
                else{
                stQuery = "SELECT " + stFieldName + " FROM " + stTableName
+ " WHERE " + stCondition;// + ";";
```

```
                }
                if(DEBUG) System.out.println(stQuery); //Debugging
                }
                return stQuery;
        }
        private String[][] fnFieldParser(String stQueryResult, int nRows, int nCols){
           String stRows[] = new String[nRows];
           String stFieldVal[][] = new String[nRows][nCols];
      int nRowIndex = 0, nColIndex  = 0;
      nRowNum = nRows;
      nColNum = nCols;
      StringTokenizer STToken1 = new StringTokenizer(stQueryResult, "%");
           while(STToken1.hasMoreTokens()){
        stRows[nRowIndex] = STToken1.nextToken();//Separating the Rows
              StringTokenizer STToken2 = new StringTokenizer(stRows[nRowIndex],
"^");
              while(STToken2.hasMoreTokens()){
                 stFieldVal[nRowIndex][nColIndex] = STToken2.nextToken();//Separating
the columns in each row
                 nColIndex = nColIndex + 1;
              }
              nColIndex = 0;
              nRowIndex = nRowIndex + 1;
           }
           return stFieldVal;
        }
}

class ApplicationServer extends Thread {
   private static final boolean DEBUG = true;
   public final static int DEFAULT_PORT = 7001;
   protected static int port=7001;
   protected ServerSocket server_port;
   protected ThreadGroup CurrentConnections;
   public ServerConnection c;
   protected Vector connections;
   protected ConnectionWatcher watcher;
   DBAdmin admin;
        static String stDBURL = "jdbc:oracle";
        static String stDBUsrName = "mariam";
   static String stDBPwd = "m15502@logic";
        static public void main(String args[]) {
      new ApplicationServer(port, stDBURL, stDBUsrName, stDBPwd);
        }
   public ApplicationServer(int port, String stDBURL, String stDBUsrName, String
stDBPwd) {
```

```
        super("Server");
        admin = new DBAdmin(stDBURL, stDBUsrName, stDBPwd);
        try { server_port = new ServerSocket(port);
                System.out.println(port);
            }
        catch (IOException e) {System.out.println(e);}
        CurrentConnections = new ThreadGroup("Server Connections");
        connections = new Vector();
        watcher = new ConnectionWatcher(this);
        this.start();
    }
    public void run() {
        try {
            while(true) {
                Socket client_socket = server_port.accept();
                c = new ServerConnection(client_socket, CurrentConnections, 3, watcher,
this, admin);
                // prevent simultaneous access.
                synchronized (connections) {
                        connections.addElement(c);
                }
            }
        }
        catch (IOException e) {System.out.println(e);}
            System.exit(0);
    }
    public String fnDisplay(){
        String tmpString="";
        tmpString = c.getInfo();
        return tmpString;
    }
    public int getNumConnections(){
        return c.numberOfConnections;
    }
}
class ServerConnection extends Thread {
    private static final boolean DEBUG = true;
    static int numberOfConnections = 0;
    protected Socket client;
    protected ConnectionWatcher watcher;
    protected DataInputStream in;
    protected PrintStream out;
    Connection con;
    public String stDBurl;
    public String stUsrName;
    public String stPwd;
```

```
  public ApplicationServer AS;
  DBAdmin admin;
  public ServerConnection(Socket client_socket, ThreadGroup CurrentConnections,
        int priority, ConnectionWatcher watcher, ApplicationServer A, DBAdmin
_admin) {
    super(CurrentConnections, "Connection number" + numberOfConnections++);
    this.setPriority(priority);
    client = client_socket;
    this.watcher = watcher;
    AS = A;
    admin = _admin;
    try {
      in = new DataInputStream(client.getInputStream());
      out = new PrintStream(client.getOutputStream());
    }
    catch (IOException e) {
          try
          {
            client.close();
            }
          catch (IOException e2)
          {
        if(DEBUG) System.err.println("Exception while getting socket streams: " + e);
        return;
      }
    }
    this.start();
  }
  public void run() {
    String inline;
    try {
      while(true) {
        inline = in.readLine();          // read in a line
            System.out.println("Printing inline"+inline);
        if (inline == null) break;
        inline=inline.trim();
        if(inline.toCharArray()[0]=='S'){
          System.out.println("\n" + this.getInfo() + " has performed a SELECT
Operation");
          String stFieldVal[][];          //The String Array that stores the results of a
query
          String stFieldName = " ";
          String stTableName = " ";
          String stCondition = " ";
          String stOutPut="";
          int ni, nj;
```

```
            out.println("DONE");
            inline = in.readLine();
            inline=inline.trim();
                        System.out.println("Printing inline 2"+inline);

            StringTokenizer st = new StringTokenizer(inline,"@");
            try{
              while(st.hasMoreTokens()){
                stFieldName = st.nextToken().trim();
                stTableName = st.nextToken().trim();
                stCondition = st.nextToken().trim();
              }
            }
            catch(Exception e){}
            if(DEBUG) System.out.println(stFieldName + stTableName + stCondition);
//Debugging
            stFieldVal =
admin.fngetFieldsDB(stFieldName.trim(),stTableName.trim(),stCondition.trim());
            for(ni=0;ni<admin.nRowNum;ni++){
              for(nj=0;nj<admin.nColNum;nj++){
                stOutPut = stOutPut + stFieldVal[ni][nj] + "^";
                      System.out.println("Query value"+stFieldVal[ni][nj]);
                      System.out.println("Query value"+stOutPut);
              }
              if(stOutPut.endsWith("^")){
                stOutPut = stOutPut.substring(0,stOutPut.length()-1);
              }
              stOutPut = stOutPut.trim() + "%";
            }
            if(stOutPut.endsWith("%")) {
              stOutPut = stOutPut.substring(0,stOutPut.length()-1);
            }
            if(DEBUG) System.out.println(stOutPut);     //Debuggging
                      System.out.println(stOutPut+"SWED");
            out.println(stOutPut.trim());
            out.println("DONE");
          }
          if(inline.toCharArray()[0]=='U'){
            System.out.println("\n" + this.getInfo() + " has performed an INSERT
Operation");
    if(DEBUG) System.out.println("Preparing to write DataBase");//Debugging
            out.println("DONE");
            inline = in.readLine();
                      System.out.println(inline);
            inline = inline.trim();
            boolean boquery = admin.fnExecSQLUpdate(inline);
```

```
                if(DEBUG) System.out.println(inline + boquery);//Debugging
                out.println("DONE");
                                }
                   out.flush();
            }
         }
         catch (IOException e) {}
         finally {
            try {
               client.close();
            }
            catch (IOException e2) {
               synchronized (watcher) {watcher.notify();}
            }
         }
      }
      public String getInfo() {
         return (client.getInetAddress().getHostName());
      }
   }
   class ConnectionWatcher extends Thread {
      protected ApplicationServer server;
      protected ConnectionWatcher(ApplicationServer s) {
         super(s.CurrentConnections, "ConnectionWatcher");
         server = s;
         this.start();
      }
      public synchronized void run() {
         while(true) {
            try {
               this.wait(10000);
            }
            catch (InterruptedException e){
               System.out.println("Caught an Interrupted Exception");
            }
            synchronized(server.connections) {
               for(int i = 0; i < server.connections.size(); i++) {
                  ServerConnection c;
                  c = (ServerConnection)server.connections.elementAt(i);
                  if (!c.isAlive()) {
                     System.out.println("\nDisconnecting from " + c.getInfo());
                     server.connections.removeElementAt(i);
                     i--;
                     c.numberOfConnections = c.numberOfConnections - 1;
                  }
               }
```

```
        }
      }
    }
}
```

Sample Source Code (Define Goals - U_FF_3.java) :

Java Applet for the Define Goals frame.

```
/**************************************************************************
<!-- REPI Web Site
<!-- Demo Version - August 14, 1998
<!-- Copyright (c) 1998
<!--
<!-- Author: Umang  Dave
<!--        Mariam  Burmawalla
<!-- Email: tanik@homer.njit.edu
**************************************************************************/
import java.applet.*;
import java.net.*;
import java.util.*;

public class U_FF_3 extends Applet{
        public String stServerName;
        public int iPortNumber;
        public String ServerReply;
        public DBClient dbclient;
      public String stdispvalue[];
      public String temp[];
      public int i;
      public int j;

      public String[] GetFromDataBase(String people_id,String goal_name) {
            stServerName = "cache.njit.edu";
            iPortNumber = 7001;
            DBClient dbclient = new DBClient(stServerName,iPortNumber);
            dbclient.ProcessCommand("S");
            ServerReply =
dbclient.ProcessCommand("GOAL_FILENAME,GOAL_DESC@goal_m@GOAL_PID
='"+people_id.trim()+"' AND GOAL_NAME='"+goal_name.trim()+"'");

            StringTokenizer sttok = new StringTokenizer(ServerReply,"^");
            stdispvalue = new String[10];
            i=0;
            while(i<10) {
                  stdispvalue[i]="\0";
```

```
                i++;
            }
            i = 0;
            while(sttok.hasMoreTokens()) {
                stdispvalue[i] = sttok.nextToken();
                j=i;
                i++;
            }
            return stdispvalue;
        }

        public void UpdateDataBase(String id,String proj_goalname,String
proj_goaldesc,String proj_goalfile) {

                temp=new String[10];
                stServerName = "cache.njit.edu";

                iPortNumber = 7001;

                DBClient dbclient = new DBClient(stServerName,iPortNumber);

            j=0;
            temp=GetFromDataBase(id,proj_goalname);
            dbclient.ProcessCommand("U");
        if(j==0)
                ServerReply = dbclient.ProcessCommand("insert into
goal_m(goal_id,goal_pid,goal_name,goal_desc,goal_filename)
values(autogoal.nextval,'"+id.trim()+"','"+proj_goalname.trim()+"','"+proj_goaldesc.trim(
)+"','"+proj_goalfile.trim()+"')");
            else
                ServerReply = dbclient.ProcessCommand("update goal_m set
goal_desc='"+proj_goaldesc.trim()+"',goal_filename='"+proj_goalfile.trim()+"' where
goal_pid='"+id.trim()+"' AND goal_name='"+proj_goalname.trim()+"'");

                System.out.println("Thisd is thz answer"+ ServerReply);
            }
}
```

Sample Source Code (Define Goals - U_FF_3.HTM) :

Define Goal front page.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2//EN">

<!-- REPI Web Site                                    -->
```

```
<!-- Demo Version - November 24, 1997                    -->
<!-- Demo Version - August 14, 1998                      -->
<!-- Copyright (c) 1997                                  -->
<!--                                                     -->
<!-- Author: Deepak  Pandit                              -->
<!--         Umang   Dave                                -->
<!--         Mariam  Burmawalla                          -->
<!-- Email: tanik@homer.njit.edu                         -->

<HTML>

<HEAD>
        <META NAME="Author" CONTENT="Deepak Pandit">
        <META HTTP-EQUIV="Content-Type" CONTENT="text/html;CHARSET=iso-
8859-1">

        <LINK REL=STYLESHEET TYPE="text/javascript" HREF="REPI.CSS"
                TITLE="Style Sheet">

        <SCRIPT LANGUAGE="JavaScript" SRC="REPI.JS"></SCRIPT>
      <SCRIPT LANGUAGE="JavaScript">
      val =new Array();
      temp = new Array();
      var templen;
      function getfromDB(id,proj_goalname) {
            ctr = 0;
            temp = document.U_FF_3.GetFromDataBase(id,proj_goalname);
            document.U_FF_3j.Goal_File.value = temp[0];
            document.U_FF_3j.U_Project_GoalDesc.value = temp[1];
      }
      function writetoDB(id,proj_goalname,proj_goaldesc,proj_goalfile) {

document.U_FF_3.UpdateDataBase(id,proj_goalname,proj_goaldesc,proj_goalfile);
      }
      function OpenFile(filename) {
            window.open(filename,"_top");
      }
      </SCRIPT>
        <TITLE>User's Fact Finding Task 3: Define Goals</TITLE>
</HEAD>
<APPLET CODE="U_FF_3.class" NAME="U_FF_3" WIDTH = 10 HEIGHT = 10>
</APPLET>
<BODY LINK="BLUE" VLINK="BLUE" ALINK="White"
BACKGROUND="REPI_BK2.JPG">
<BIG><CENTER><STRONG>
```

```
            <FONT COLOR="Black" CLASS="TaskTitleFormat">Task 3: Define
Goals</FONT>
</STRONG></CENTER></BIG>
<CENTER><IMG SRC="REPI_LN1.GIF" WIDTH="800" HEIGHT="5"></CENTER>

<CENTER>
<FORM NAME="U_FF_3j" METHOD="GET">
<P>
 From your point of view, list the goals to be reached by this project.
</P>
<TABLE WIDTH="100%" BORDER="0">
    <TR>
        <TH ALIGN="RIGHT">USER ID:</TH>
        <TD>
            <INPUT TYPE="TEXT" SIZE="11" NAME="Stakeholder_Info_ID">
        </TD>
    </TR>
      <TR>
            <TD ALIGN="right">
                <STRONG>Project Goal Name:  </STRONG>
            </TD>
            <TD>
                <INPUT TYPE="text" NAME="U_Project_GoalName"
SIZE="40">
            </TD>
      </TR>
    <TR><TD COLSPAN="2"> </TD></TR>
    <TR><TD COLSPAN="2"> </TD></TR>
    <TR><TD COLSPAN="2"> </TD></TR>
    <TR>
        <TH ALIGN="RIGHT">Goal File :</TH>
        <TD>
        <INPUT TYPE="TEXT" SIZE="20" NAME="Goal_File">
        </TD>
        <TD>
        <INPUT TYPE="BUTTON" NAME="U_FF_3_File"
            VALUE="Open Document"
            onClick = "OpenFile(Goal_File.value)">
        </TD>
    </TR>
</TABLE>
<BR>
<TABLE>
    <TR>
        <TH ALIGN="Centre">OR</TH>
    </TR>
```

```
</TABLE>
<TABLE>
    <TR>
        <TH ALIGN="Centre">Goal Description :</TH>
    </TR>
    <TR>
        <TD>
        <INPUT TYPE="TEXT" ALIGN="RIGHT" SIZE="80"
NAME="U_Project_GoalDesc">
        </TD>
    </TR>
</TABLE>
<BR><BR>
<TABLE WIDTH="100%" BORDER="0">
      <TR>
            <TD ALIGN="Center">
                  <INPUT TYPE="Button" NAME="U_FF_3_Enter"
                        VALUE="Enter Information"
   onClick =
"writetoDB(Stakeholder_Info_ID.value,U_Project_GoalName.value,U_Project_GoalDes
c.value,Goal_File.value)">
            </TD>
        <TD ALIGN="Center" COLSPAN="2">
            <INPUT TYPE="Button" NAME="U_FF_1_Get"
                  VALUE="Get Information" onClick =
"getfromDB(Stakeholder_Info_ID.value,U_Project_GoalName.value)">
        </TD>
            <TD ALIGN="Center">
                  <INPUT TYPE="Reset" VALUE="Clear Form">
            </TD>
      </TR>
</TABLE>
</FORM>
</CENTER>
</BODY>
</HTML>
```

The requirement elicitation process in an iterative process. According to the framework provided by the Software Engineering Institute the users as well as the developers are guided to iterate through the five phases of Fact Finding, Gathering and Classification, Evaluation and Rationalization, Prioritization and Planning, and Integration and Validation. The end result of this iterative process would be well-understood requirements.

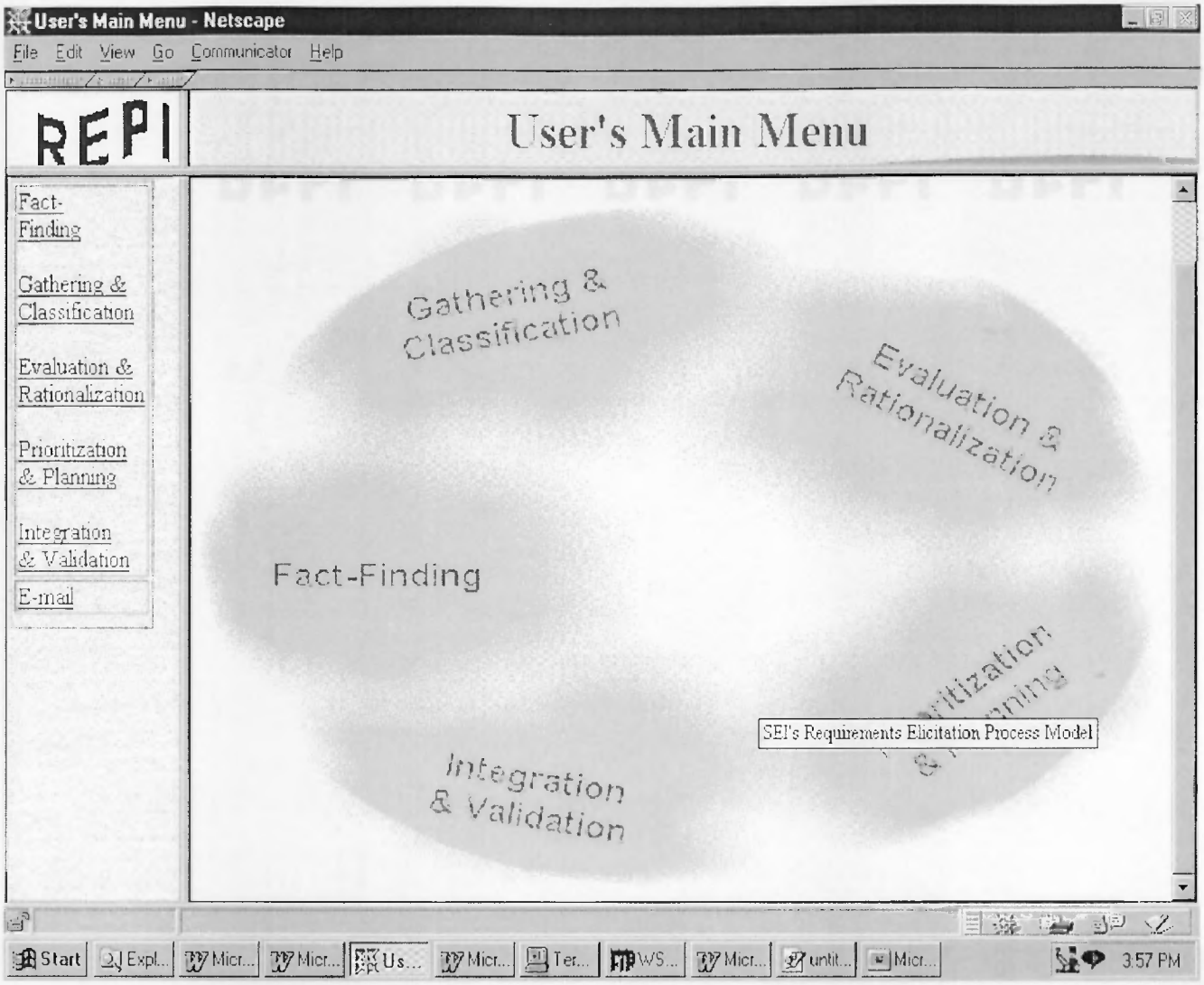

Figure 9 Phases of Requirement Elicitation Process on the Internet

1. ***Fact Finding Phase:*** The Fact Finding phase comprises of identifying relevant people on the user side, and identifying domain experts on the developer side. This phase allows the user to describe the problem, define the goals and list mission scenarios. The developer on the other hand identifies domain and architectural models, conducts technological surveys and assesses constraints.

a) Identify Relevant People / Identify Domain Experts :

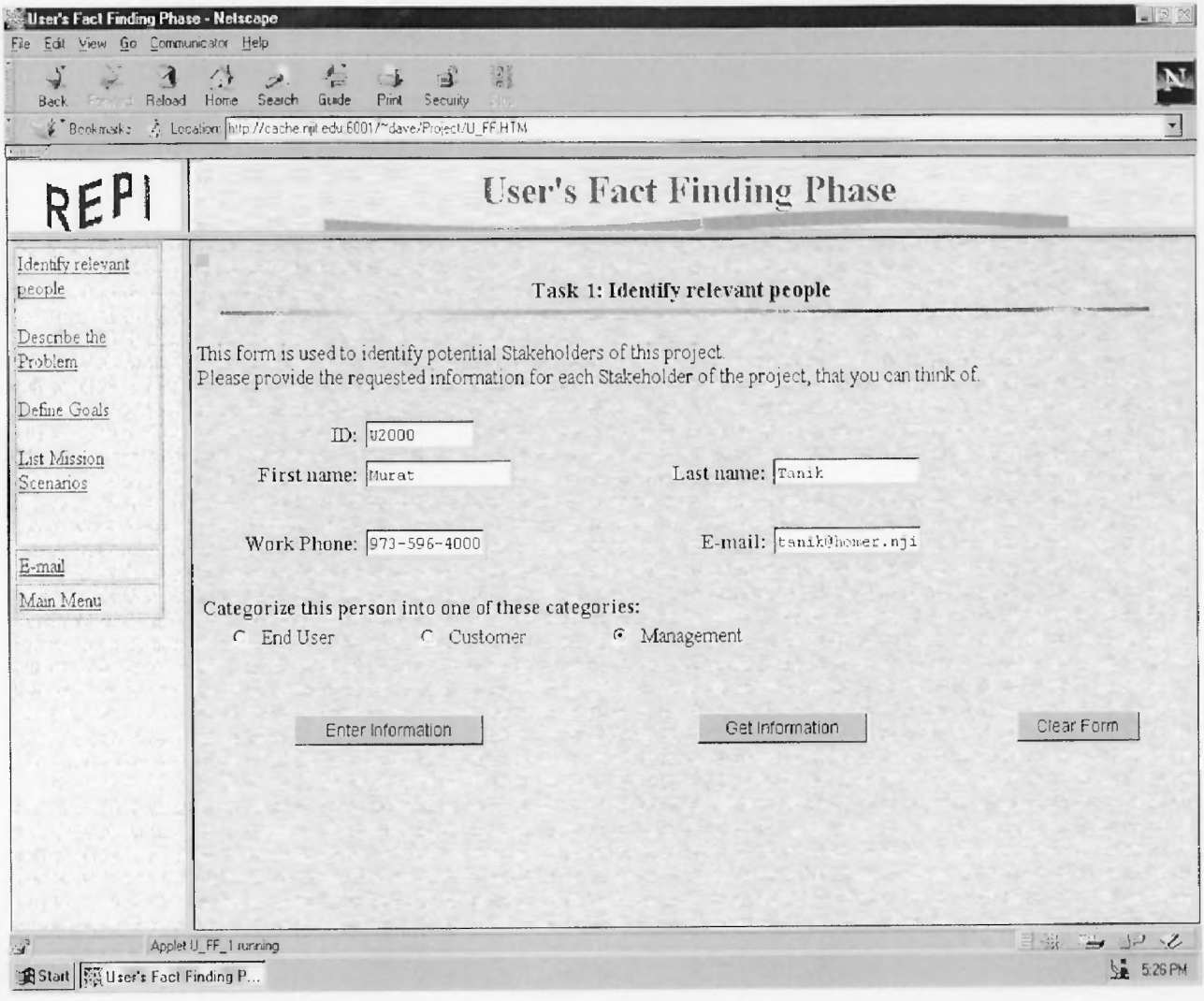

Figure 10 Identify Relevant People / Identify Domain Experts

Source Code (Identify Relevant People - U_FF_1.java) :

```
/*************************************************************************
<!-- REPI Web Site
<!-- Demo Version - August 14, 1998
<!-- Copyright (c) 1997
<!--
<!--        Umang  Dave
<!--        Mariam  Burmawalla
<!-- Email: tanik@homer.njit.edu
*************************************************************************/
import java.applet.*;
import java.net.*;
import java.util.*;

public class U_FF_1 extends Applet{
        public String stServerName;
        public int iPortNumber;
        public String ServerReply;
        public DBClient dbclient;
        public String stdispvalue[];
        public String temp[];
        public int i;
        public int j;

        public String[] GetFromDataBase(String people_id) {
                stServerName = "cache.njit.edu";
           iPortNumber = 7001;
           DBClient dbclient = new DBClient(stServerName,iPortNumber);
                dbclient.ProcessCommand("S");
                ServerReply =
dbclient.ProcessCommand("PEOPLE_FNAME,PEOPLE_LNAME,PEOPLE_EMAIL,P
EOPLE_PHONE,PEOPLE_TYPE@people_m@PEOPLE_ID= '"+people_id.trim()+"'");
                System.out.println("Thisd is thz answer"+ ServerReply);
                StringTokenizer sttok = new StringTokenizer(ServerReply,"^");
                stdispvalue = new String[10];
                i=0;
                while(i<10) {
                        stdispvalue[i]="\0";
                        i++;
                }
                i = 0;
                while(sttok.hasMoreTokens()) {
                        stdispvalue[i] = sttok.nextToken();
                        System.out.println("Token is"+stdispvalue[i]);
                        j=i;
```

```
                i++;
            }
            System.out.println("Token is asd`"+stdispvalue);
            return stdispvalue;
        }

        public void updatePage() {
            StringTokenizer sttok = new StringTokenizer(ServerReply,"^");
            while(sttok.hasMoreTokens()) {
                System.out.println("Token is"+sttok.nextToken());
            }
        }

        public void UpdateDataBase(String people_id, String people_fname, String
people_lname, String people_email,String people_phone, String people_type,String cat) {

            temp = new String[10];
            stServerName = "cache.njit.edu";

            iPortNumber = 7001;

            DBClient dbclient = new DBClient(stServerName,iPortNumber);

            j=0;
            temp=GetFromDataBase(people_id);
            dbclient.ProcessCommand("U");
        if(j==0)
        {
            ServerReply = dbclient.ProcessCommand("insert into
people_m(people_id,people_fname,people_lname,people_email,people_phone,people_ty
pe,people_category)
values('"+people_id.trim()+"','"+people_fname.trim()+"','"+people_lname.trim()+"','"+pe
ople_email.trim()+"','"+people_phone.trim()+"','"+people_type.trim()+"','"+cat.trim()+"')"
);
        }
        else
            ServerReply = dbclient.ProcessCommand("update people_m set
people_fname='"+people_fname.trim()+"',people_lname='"+people_lname.trim()+"',peop
le_email='"+people_email.trim()+"',people_phone='"+people_phone.trim()+"',people_typ
e='"+people_type.trim()+"' where people_id='"+people_id.trim()+"'");
            System.out.println("Thisd is thz answer"+ ServerReply);
        }
}
```

b) Describe the problem: From the viewpoint of the stakeholder, describe the problem that is to be solved by this project.

c) Define Goals: From the viewpoint of the stakeholder, list the goals to be reached by this project.

d) List Mission Scenarios: This scenario will describe how the development team will approach the problem of designing the system.

e) Identify Domain Models: Identify the domain models and architectural models to be used by this project.

f) Conduct Technological Survey: Enter the survey information.

g) Assess Constraints: Enter constraint information.

2. ***Gathering and Classification Phase:*** The Gathering and Classification phase comprises of gathering requirements, listing requirements entered by various stakeholder and classifying requirements as per their category.

a) Add Requirements: Many engineers are not trained to enter requirements, this leads to a major problem of stating design implementation. This may occur if a customer mandates a design solution as a requirement or the developer generates derived requirements, which actually may be design solutions. The dangers in stating implementation are forcing a design when not intended and the author may be lulled into believing that all requirements have been covered, even though some important requirements are missing. This leads to not delivering a product, which accomplishes the customer's real intent.

The requirement must be complete to provide all project specific information to the stakeholders. The requirement must be unambiguous so that different users would give same interpretation to the requirement. Moreover, it should also avoid the ambiguity arising from the lack of details and must be properly scoped. The requirement must be concise in manner.

Requirement Compliance level defines compliance level like :

- Mandatory: Even if the requirement is primary requirement or a derived requirement it must be implemented.
- Guidance: It provides some level of guidance towards the implementation of the system. Failure to implement does not mean noncompliance, if reasonable degree of implementation was attempted.
- Information: These are not actually requirements, but are non-binding statements, which significantly influence the project development.

The stakeholders to classify requirements into groups use requirement Categories. The classification can be into groups that can address a particular goal.

The current status of the requirements can be :

- To be Defined: Such requirements do not have defined values.
- To be reviewed: Such requirements needs further review.
- Defined: Final value of the requirement has been obtained.
- Approved: The requirement has been reviewed and approved.
- Verified: The requirement has been verified with verification plan.
- Deleted: The requirement is no longer applicable.

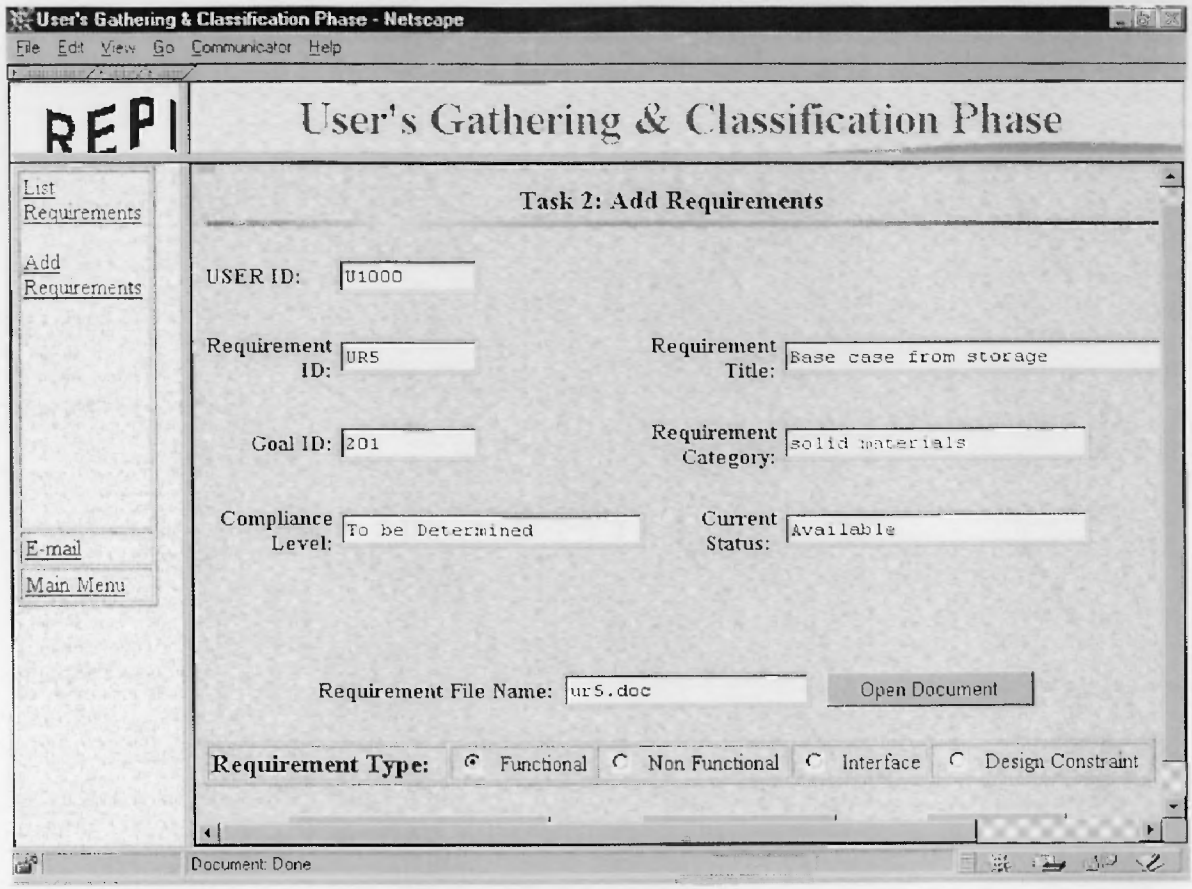

Figure 11 Add Requirements

Source Code (Add Requirements - U_GC_2.java) :

```
/**************************************************************************
<!-- REPI Web Site
<!-- Demo Version - August 14, 1998
<!-- Copyright (c) 1998
<!--
<!-- Author: Umang  Dave
<!--         Mariam  Burmawalla
<!-- Email: tanik@homer.njit.edu
**************************************************************************/
import java.applet.*;
import java.net.*;
import java.util.*;

public class U_GC_2 extends Applet{
        public String stServerName;
```

```
    public int iPortNumber;
    public String ServerReply;
    public DBClient dbclient;
  public String stdispvalue[];
  public String temp[];
  public int i;
  public int j;


    public String[] GetFromDataBase(String id) {
          stServerName = "cache.njit.edu";
      iPortNumber = 7001;
      DBClient dbclient = new DBClient(stServerName,iPortNumber);
          dbclient.ProcessCommand("S");

          ServerReply =
dbclient.ProcessCommand("req_pid,req_title,req_gid,req_category,req_complevel,req_st
atus,req_filename@req_m@req_id= '"+id.trim()+"'");
          StringTokenizer sttok = new StringTokenizer(ServerReply,"^");
      stdispvalue = new String[10];
      i=0;
      while(i<10) {
          stdispvalue[i]="\0";
          i++;
      }
      i = 0;
      while(sttok.hasMoreTokens()) {
          stdispvalue[i] = sttok.nextToken();
          j=i;
          i++;
      }
      return stdispvalue;
  }

    public void updatePage() {
          StringTokenizer sttok = new StringTokenizer(ServerReply,"^");
          while(sttok.hasMoreTokens()) {
              System.out.println("Token is"+sttok.nextToken());
          }
  }

    public void UpdateDataBase(String id,String req_id, String req_title, String gid,
String cat,String complvl, String stat, String type,String filename) {

          stServerName = "cache.njit.edu";
```

```
                iPortNumber = 7001;

                DBClient dbclient = new DBClient(stServerName,iPortNumber);

            j=0;
            temp=GetFromDataBase(req_id);
            dbclient.ProcessCommand("U");
        if(j==0)
                ServerReply = dbclient.ProcessCommand("insert into
req_m(req_pid,req_id,req_title,req_gid,req_category,req_complevel,req_status,req_type,r
eq_filename,req_dummy,req_valid)
values('"+id.trim()+"','"+req_id.trim()+"','"+req_title.trim()+"','"+gid.trim()+"','"+cat.trim(
)+"','"+complvl.trim()+"','"+stat.trim()+"','"+type.trim()+"','"+filename.trim()+"','NA','N')
");
            else
                ServerReply = dbclient.ProcessCommand("update req_m set
req_pid='"+id.trim()+"',req_title='"+req_title.trim()+"',req_gid='"+gid.trim()+"',req_categ
ory='"+cat.trim()+"',req_complevel='"+complvl.trim()+"',req_status='"+stat.trim()+"',req
_type='"+type.trim()+"',req_filename='"+filename.trim()+"' where
req_id='"+req_id.trim()+"'");
            }
}
```

b) List Requirements: List the entire set of requirements or requirements specific to a particular user.

c) Classify Requirements: Classify the requirements as per their category.

*3. **Evaluation and Rationalization Phase:*** Perform abstraction and derive rationale from the entered requirements.

a) Perform Abstraction: To ensure that a requirement statement represents a need and not an implementation, ask WHY the requirement is needed. If this does not lead to a "real" need statement, then the original requirement statement is probably appropriate, citing a need rather than implementation [HARWELL 93].

b) Capture Rationale: This attribute provides a link to the data or tradeoff analysis, which support the requirement. The supporting data may include the reason or reasons a requirement is needed. This attribute provides an index to the rationale for every requirement [KAR 96].

c) Risk Assessment: This attribute provides a record of risk associated with a requirement, if nay. A quantitative assessment of the risk is provided.

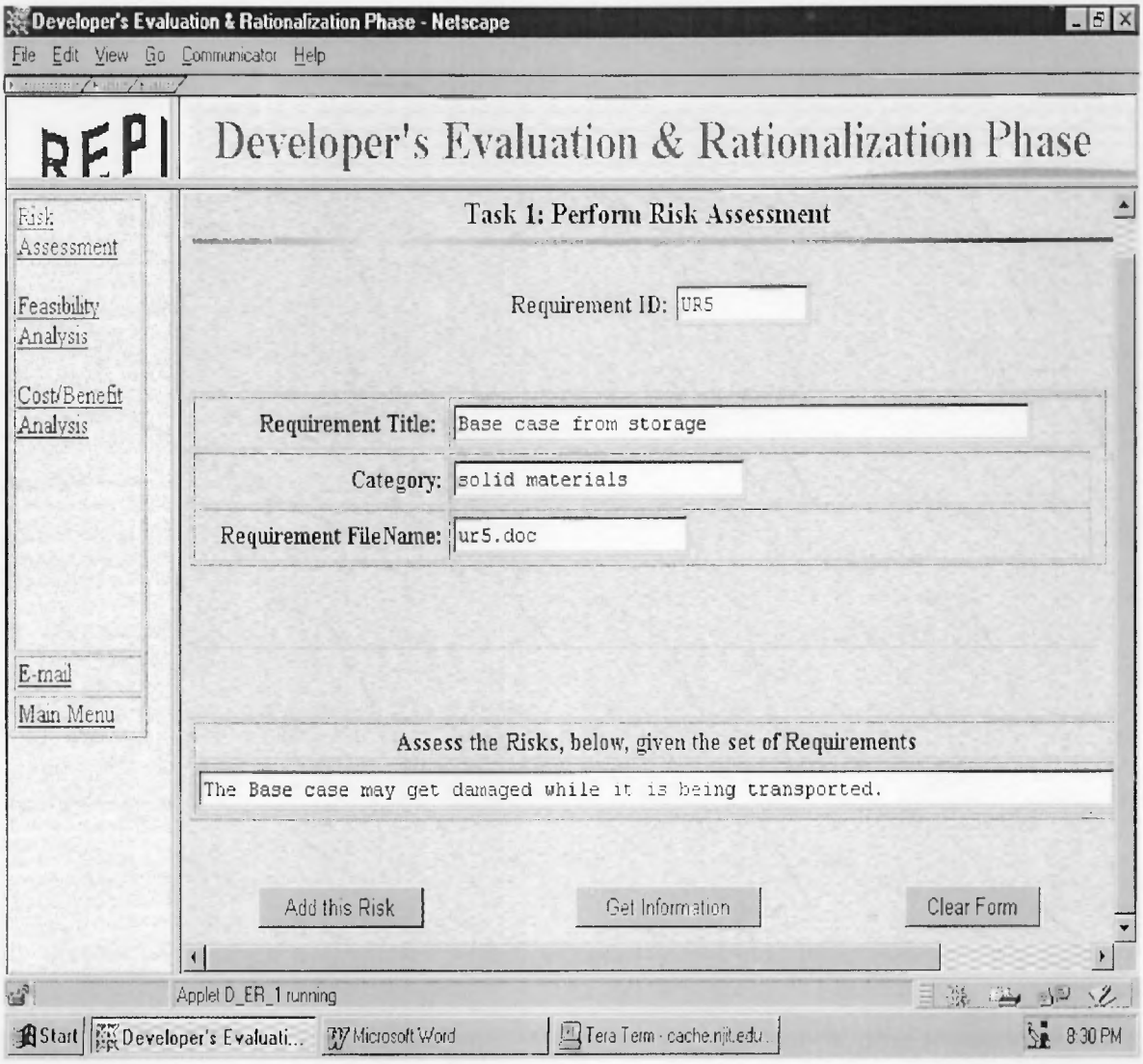

Figure 12 Perform Risk Assessment.

d) Feasibility Analysis: This attribute specifies if the stated requirement can be achieved by one or more developed system concepts at a definable cost. Various levels of studies are included to measure then feasibility of a requirement.

d) Cost / Benefit Analysis: This attribute specifies if the given requirement is cost effective and if it beneficial to the developer, the user and the product at large.

*4. **Prioritization and Planning Phase:*** Prioritize the given set of requirements and plan the incremental stages depending on the priorities.

a) Prioritize Requirements: This characteristic identifies the relative importance of a requirement in terms of implementation, particularly in establishing criteria for trade studies. To mandate that certain elements must be completed before a specified ceiling is reached. The priority characteristic may also be used for establishing the sequence in which specified design or test activities should occur [HARWELL 93].

Some of the parameters that can influence the prioritization are : cost level, dependence level, level of understanding, Importance level, priority.

b) Plan Incremental Development Stages: This attribute helps is planning the development stages once the priority of all the requirements have been established. This helps the developer approach the design phase in a structured manner.

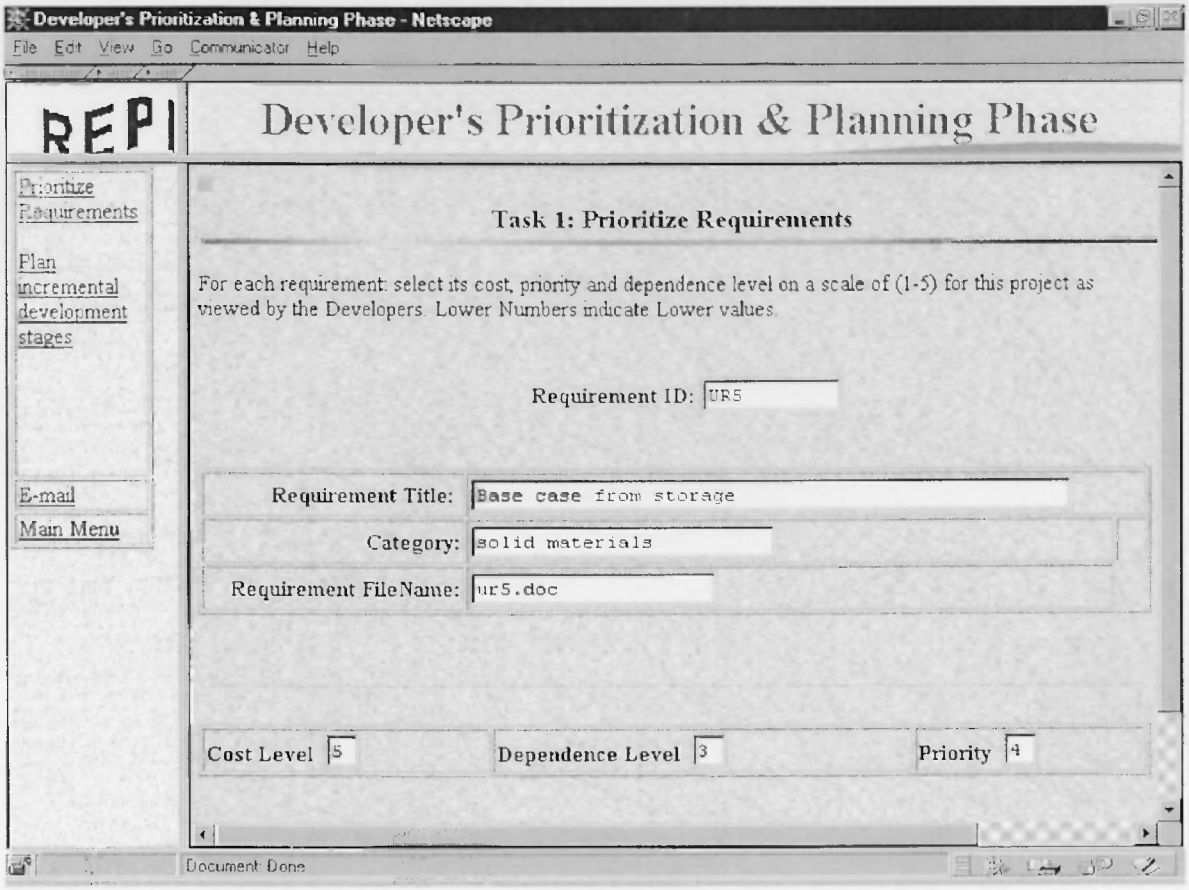

Figure 13 Prioritization and Planning.

5. Integration and Validation Phase: This phase comprises of verifying the requirements for correctness and completeness and finally authorizing well defined requirements.

a) Validate Requirements: This attribute checks the requirement for its completeness in the sense that it provides all the program-specific information necessary for the user to carry out the nest process step. Moreover, it validates the requirement for a particular goal.

The stated requirement does not contradict other requirements and it is not a duplicate of another requirement.

In a large set of requirements, which are not well organized by some clearly defined categories, it may be hard to spot duplications and inconsistencies. It is therefore important to organize the requirements so inconsistencies can be spotted during the reviews [KAR 96].

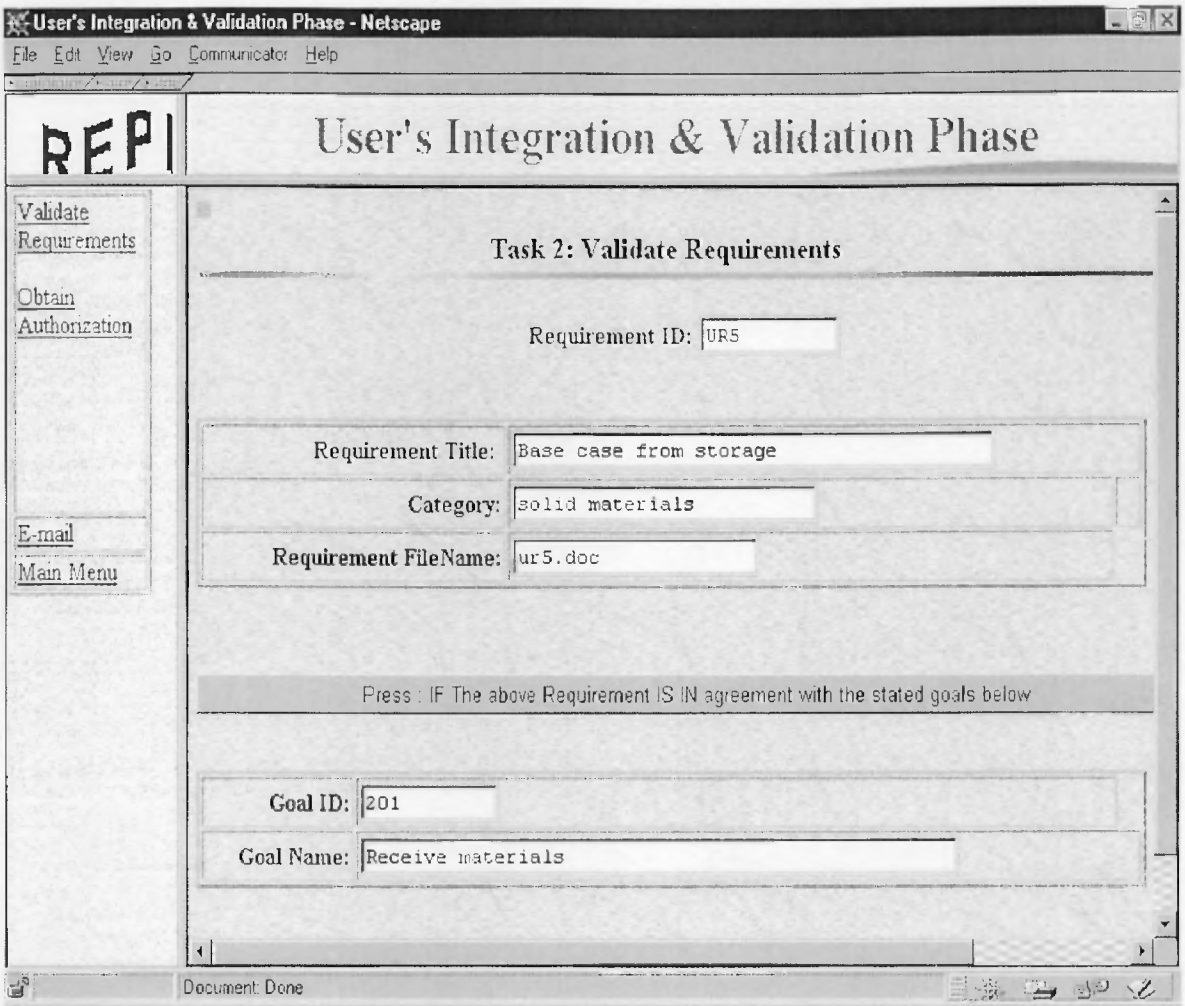

Figure 14 Integration and Validation

b) Obtain Authorization: Every requirement is authorized after confirming that it is not vague or general but is quantifies. The requirement is authorized after inspection, analysis, demonstration or test.

CHAPTER 5

CONCLUSION AND FUTURE WORK

The thesis is aimed at utilizing client server, relational database management systems and Internet technologies for the purpose of Requirement Elicitation. Chapter 1 introduces the concepts of knowledge management, requirement engineering and environmental life cycle. Chapter 2 explores the concepts in detail addressing the issues related to knowledge management like knowledge representation, filtering and knowledge search. It also describes Software Engineering Institute's Requirement Elicitation framework, process model, techniques, methods and methodologies. Chapter 3 describes the problems facing the domains of requirement engineering, knowledge management and environmental life cycle. Chapter 4 describes one approach using client server, relational database, and Internet technologies for effective requirement elicitation. It also explains in detail how these technologies can facilitate efficient communication and information sharing between stakeholders and developers. These technologies can also help them arrive at a common understanding.

5.1 Benefits of Requirement Elicitation Web Presence

The accurate translation of a customer's need by the product team is essential. Identifying requirements characteristics, determining that the stated requirement conveys a need and ascertaining contextual adequacy are essential for assessing the accuracy of translation before the product team proceeds. These criteria enable the analysis team to better understand the customers intent and to better identify where disjoints, ambiguities, and conflicts exist early in the program. The objective is to be able to integrate a collection of

requirements and to establish their achievability prior to design implementation – rather then during test and evaluation. In addition, as we learn the effects certain combinations of characteristics and relationships have on the development process, we can more readily identify potential difficulties and correct them before we are committed to an approach that maybe impracticable.[HARWELL 93].

Using the web as the platform, "facilitates the distribution of the application and its data to geographically-separated users on diverse computing platforms" [GIRGENSOHN 96]. One of the major benefits of requirement elicitation on web is that it allows people and organizations separated in space of time to exchange information and come to consensus on the needs of the people. The REPI web site provides a distributed asynchronous environment allowing group members to interact at different time from different places. The advantages of such meetings is that "group members do not have to be physically in the same place to meet, not must they communicate with one another at the same time" [OCKER 95]. These two characteristics of distributed asynchronous communication extend the definition of a meeting; this expanded definition of a meeting loosens the constraints in an organization and thus increases the means by which groups can accomplish their work [OCKER 95].

"Organizational and social issues have great influence on the effectiveness of communication activities" and therefore on the overall success or failure of a given project [AL-RAWAS 96]. Expensive communication channel can seriously hamper the exchange of information between the group members. Sometimes surrogates or intermediaries are used as a representative to communicate with the client or developer side instead of actual clients or developers. Such indirect links of communication can

lead to misunderstood requirements. [KEIL 95] reports that direct links are better than indirect links because intermediaries might filter or distort messages between the two groups and they might not have a complete understanding of the customer's needs. [KEIL 95] also reports that up to a certain point the more links between customer's and developers, the better it is for the development processes. Another possible limitation of the expensive communication channel is that it might be restricted to one way communication. The development side people might produce documents based on their understanding and send these voluminous documents for the client side. They might not take the time to validate the requirements, even if they understand the notations used in the specifications. All these limitations of expensive communication channel reduce the accuracy of the information [AL-RAWAS 96].

The requirement elicitation process model addresses complexity and volatility issues present in the elicitation process. Software products are orders of magnitude more complex than other human constructs and conceiving, describing, and testing these products is difficult [BROOKS 87]. Additionally, the computational and memory capacities of humans are limited and can server as a hindrance during the creation of complex requirements. These factors are solved by the iterative nature of the model. Iteration has been identified as a suitable technique for addressing change and complexity [MILLER 93]. The process model takes into consideration the importance of including all the stakeholders in the definition of the tasks and responsibilities. This is necessary for the requirements elicitation process to be successful [CHRISTEL 92].

The requirement elicitation process on the web provides database (Oracle 7) connectivity. Providing a form of database connectivity in the back end helps in storing

all the information related to the requirements. This information can be very useful and can be readily available in the later stages of design and implementation. The information present in the database can also very used to generate reports and presentations.

The first phase of the process model is the fact-finding phase. For this phase the accuracy and completeness is critical to the success of the entire process. Multiple passes through this stage should be considered to promote completeness [MILLER 93]. This phase promotes a feeling of importance among participants and forms a forum for clarification and understanding and fostering sense of commitment.

During the gathering and classification phase it is important to decompose the requirements into manageable pieces. If multiple iterations of the entire requirements elicitation process are required, the decomposition will aid in the task of incremental application and development of the requirements [CHRISTEL 92]. It entails the formalization of documents and models required in achieving the goals. It provides a mechanism for representing various types of requirements and integrating them.

The benefit of the evaluation and the rationalization phase is that it provides the missing rationale for the requirements gathered in the fact-finding and gathering and classification phase. Moreover, it also provides an analysis of the risk, cost / benefits, feasibility and why the requirements are required.

Prioritization and planning phase helps in arranging the requirements in order of relative importance from the view of the client and the view of the developer [CHRISTEL 92]. This allows the developers to focus on the most important issues and then arranging the rest of the requirements in their relative importance. It also clearly explains what exactly the customer is looking for in a system.

Integration and validation phase ensures completeness, conflicts, quality requirements and reduces conflicts. This phase also identifies incomplete requirements and allows further iteration.

Two evaluation criteria for requirements capture methodologies are [CHRISTEL 92]:

1. Effectiveness, the achievement of the highest valued goals; and
2. Efficiency, achieving goals without consuming more resources than necessary.

5.2 Future Work

The requirement elicitation web site developed for this thesis describes one approach for requirement elicitation. Future work on the project will improve its usefulness and effectiveness. This section describes the major areas in which future work is possible.

One of the major areas of research would be to add artificial intelligence to the existing tool. This can prompt the user while using the tool and can guide him / her in the proper direction. It can also guide in situations where the user in entering some design issues or implementation issues instead of requirements.

The tool currently available is a generic model, it could be expanded to facilitate functions specific to the application. Moreover, features like image processing and video conferencing can be add to the functionality. The tool can also be made expanded to provide information of the currently logged group members and can initiate a face-to-face meeting. An image of the group member can be displayed along with the requirement specified. In this way all the group members can know each other better.

For the fact-finding phase if the understanding and the representation of the problem domain is mature, the objectives of this phase may be easily identified,

understood, and completed. However, if this is not the case, cross-functional teams should be engaged to perform this task [MILLER 93]. Information may not be gathered or documented correctly and could be filtered.

Gathering and classification phase introduces redundant data if information must be reentered from the other representatives. This creates additional work to create and maintain the model [MILLER 93]. Processes can be added to this phase to check the redundancy and reduce the maintenance cost.

In, evaluation and rationalization phase little documentation is available to support how the models and representations should be compared and contrasted in order to evaluate them successfully [MILLER 93]. Employing different representation techniques can lead to difficulty in comparing even the complete models. The techniques for evaluating risk, cost / benefits and feasibility are limited. Moreover, these techniques may vary from person to person, this can lead to some difficulty in the rationalization phase.

In the prioritization and planning phase it is very important that the group of people who are fully aware of the requirements and the functionality of the system must execute the planning. Allowing all the group members to participate in planning can hinder the system design. Proper review and prioritization are beneficial before proceeding to the planning.

The functionality of drop down menus for the items to be selected from the database can be added. Moreover, as a benchmark case object oriented databases can be used to storing information. The object oriented feature can add to the functionality and effectiveness of the tool.

To summarize, the web site can be improved by incorporating the above functionality. The basics of well defined requirements do not require elegant, entertaining prose. In the words of Albert Einstein, "When you are out to describe the truth, leave elegance to the tailor."

REFERENCES

[AL-RAWAS 96] AL-Rawas, Amer and Easterbrook, Steve. "Communication Problems in Requirements Engineering : A Field Study", in *Proceedings of the First Westminster Conference on Professional Awareness in Software Engineering. 1996.*

[BERLIN 89] Berlin, Lucy M. "User-Centered Application Definition: A Methodology and Case Study", *Hewlett-Packard Journal 40(5): pp. 90-97, October 1989.*

[BRACKETT 90] Brackett, John W. "Software Requirements", *SEI Curriculum Module SEI-CM-191.2. Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA. January 1990.* http://www.sei.cmu.edu/publications/documents/cms/cm.019.html (20 July, 1998)

[BROOKS 87] Brooks, F. P. Jr. No Silver Bullet : "Essence and Accidents of Software Engineering", *IEEE Computer, pp. 10-19, April 1987.*

[CHRISTEL 92] Christel, Michael G. and Kang, Kyo C. "Issues in Requirement Elicitation.", *Technical Report CNW/SEI-02-TR-12 or ESC-TR-92-012. Software Engineering Institute, Carnegie Mellon University, Pittsburgh PA. September 1992.* http://www.sei.cmu.edu/publications/documents/92.reports/92.tr.012.html (20 July, 1998)

[DAVIS 93] Davis, Alan M. Software Requirements: Objects, Functions and States, P T R Prentice Hall, Englewood Cliffs, NJ. 1993

[DANIEL 98] Daniel E. O'Leary. "Enterprise Knowledge Management", *IEEE Computer, pp. 54-61, March 1998.*

[DEBENHAM 97] John Debenham. "Constraints of knowledge Management", AAAI Spring Symposium, Artificial Intelligence in Knowledge Management Stanford University. March 24-26, 1997. http://ksi.cpsc.ucalgary.ca/AIKM97/debenham/debenham.html (13 August, 1998).

[ELMASRI 94] Elmasri / Navathe. Fundamentals of Database Systems, Second Edition, Addison-Wesley Publishing Company, 2725 Sand Hill Road, Menlo Park, CA 94025. 1994.

[GIRGENSOHN 96] Girgensohn, Andread. "Experiences in Developing Collaborative Applications Using the World Wide Web 'Shell'." Hypertext 96 : The Seventh ACM Conference on Hypertext. 1996.

[GRAEDEL 95] T. E. Graedel, B. R- Allenby. Industrial Ecology, Prentice Hall, Englewood Cliffs, New Jersey 07632. 1995.

[HARWELL 93] Harwell, Richard. "What Is A Requirement?", *Published in the Proceedings of the third International Symposium of the INCOSE, 1993.*

[HERLEA 97] Herlea, Daniela. "Knowledge Management for Requirement Engineering", AAAI Spring Symposium, Artificial Intelligence in Knowledge Management Stanford University. March 24-26, 1997. http://ksi.cpsc.ucalgary.ca/AIKM97/herlea/KMSE.html (13 August, 1998)

[IVY 90] Ivy Hooks. "WHY JOHNNY CAN'T WRITE REQUIREMENTS", *Paper given at AIAA conference, 1990.*

[KAR 96] Kar, Pradip and Bailey, Michelle. "Characteristics of Good Requirements', *Paper given at the 6th INCOSE Symposium, 1996.*

[KEIL 95] Keil, Mark and Carmel, Erran. "Customer-Developer Links in Software Development", Communications of the ACM. Volume 38, Number 5. May 1995.

[MILLER 93] Miller, U Greg and Tanik, Murat M. "Multimedia Applications in Software Engineering", *Technical report 93-CSE-50. Southern Methodist University, Dallas, Texas. November 1993.*

[OBJA 95] Object Agency Semantic Networks for Object Oriented software engineering.

[OCKER 95] Ocker, Rosalie, Hiltz, Starr Roxanne, et al. "The effects of distributed group support and process structuring on software requirements." Journal of Management Information Systems. Winter 95/96, Volume 12, Issue 3, 1995.

[ORACLE 96] "Oracle Intranet Strategy", An Oracle White Paper. Oracle Corporation, Redwood Shores, CA- July 1996. http://www.oracle.com/promotions/intranet/html/intrane_wp.html (14 Feb, 1998)

[ORACLE 7] Oracle 7 Server, SQL language reference manual.

[PEYMAN 97] Peyman Zehtab-Fard. "Knowledge Representation for Software Development", *Submission to Umea's first Students Conference in Computing Sciences, 1997.* http://www.cs.umu.se/~dvlpzd/extended.html (25 July, 1998).

[PLAYLE 96] Playle, Greg and Schroeder, Charles. "Software Requirements Elicitation Problems, Tools, and Techniques". In CrossTalk – The Journal of Defense Software Engineering, 1996. http://www.stsc.hill.af.mil/crosstalk/1996/dec/xt96d12e.html (3 March, 1998).

[POHL 93] Pohl, Klaus. "The Three Dimensions of Requirement Engineering", *Technical Report NATURE-92-11. 5th International conference on Advanced Information Systems Engineering, Paris, France. June 1993.*

[RAGHAVAN 94] Raghaven, Sridhar, Zelesnik, Gregory, and Ford, Gray. Lecture Notes on "Requirements Elicitation", *Report CMU/SEI-94-EM-10. Software Engineering Institute, Carnegie Melon University, Pittsburgh PA. 1994.* http://www.sei.cmu.edu/publications/documents/ems/94.em.010.html (25 July, 1998).

[RZEPKA 89] Rzepka, William E. "A Requirements Engineering Testbed: Concept, Status, and First Results". *In Bruce D. Shriver (editor), Proceedings of the Twenty-Second Annual Hawaii International Conference on System Sciences, 339-347. IEEE Computer Society, 1989.*

[SKUCE 97] Skuce, Doug. "Hybrid KM: Integrating Documents, Knowledge Bases, Databases, and the Web", AAAI Spring Symposium, Artificial Intelligence in Knowledge Management Stanford University. March 24-26, 1997. http://ksi.cpsc.ucalgary.ca:80/AIKM97/skuce/skuce.html (13 August, 1998)

[SUN 98] "The Java™ Language: An overview", Sun Microsystems, Inc. http://java.sun.com:80/docs/Overviews/java/java-overview-1.html (13 August, 1998).